MASTER EN SCIENCES INFORMATIQUES

Microscopie virtuelle
amélioration d'une plate-forme de microscopie virtuelle

Bocken, Xavier

Award date:
2008

Institut d'Informatique
FUNDP - Namur
Rue Grandgagnage, 21
B-5000 Namur

Cliniques Universitaires
UCL - Mont-Godinne
Avenue du Docteur Gaston Thérasse, 1
B-5530 Yvoir

Microscopie virtuelle

Amélioration d'une plate-forme de microscopie virtuelle

Xavier Bocken

Promoteur : Jean-Paul Leclercq
Co-Promoteur : Hubert Meurisse

Année académique 2007-2008

Mémoire présenté en vue de l'obtention du grade de Maître en Informatique

Résumé

L'informatique s'est installée dans tous les domaines et la médecine n'y a pas échappé. Le développement des technologies médicales ont fait éclore un nouveau domaine appelé télémédecine. Ce domaine a permis, depuis de nombreuses années, à des étudiants de travailler au développement de diverses applications de support aux activités du laboratoire d'hématologie de Mont-Godinne. Ces applications portaient sur la numérisation de frottis sanguins et leur traitement afin de fournir une aide au diagnostic. D'autres développements visaient à partager l'information permettant un diagnostic en vue de créer des archives ou simplement de recueillir des avis multiples.

Ce mémoire propose des améliorations aux différents développements et tente d'apporter des solutions à certains problèmes persistants. Ainsi, un problème récurrent est lié à l'autofocus monté sur le microscope d'acquisition. De plus, ce mémoire expose les modifications qui ont été apportées à l'application de coregistration qui intervient après la capture des images pour créer avec cet ensemble une méga-image. Dans la continuité de cette automatisation, il présente certaines avancées dans la caractérisation des globules blancs sur base de leurs caractéristiques visuelles. Dans une autre optique, ce mémoire présente aussi une application d'extraction et d'envoi de données numériques issues d'analyses.

Mots-Clés : télémédecine, télémicroscopie, microscopie virtuelle, imagerie médicale, frotti numérique, caractérisation, globules blancs, classification, DICOM, mega-image, coregistration

Abstract

In present times, computer sciences touch all sectors of human life, health cares and medicine are also affected. The development of medical technologies has shown a new domain which is named telemedicine. This gives, for many years, opportunities to students for working on applications supporting activities of the laboratory of hematology in the University Clinics of Mont Godinne. These applications work on numerisation of blood smears and their treatment for giving help to diagnosis. The aim of other developments is to share information with the idea to backup it or to get additional expert opinions.

This master thesis proposes some improvements to past developments and try to solve some persistant issues. So a first problem is due to errors generated by the autofocus on the acquisition microscope. Moreover, this thesis exposes modifications applied to the coregistration application which is active after the capture of all pictures for creating with the whole set of picture a mega-image. For continuing the automation, it presents also some advances in caracterisation of white blood cells based on visual features. By another way, this document presents also an application which can extract and send numerical data coming from analysis.

Keywords : telemedicine, telemicroscopy, virtual microscopy, medical imagery, numerical smear, caracterisation, white blood cells, classification, DICOM, mega-image, coregistration

Remerciements

Tout au long de mon stage et de l'écriture de ce mémoire, j'ai pu compter sur l'appui de nombreuses personnes et je souhaiterais leur adresser mes sincères remerciements.

Je remercie tout d'abord Monsieur **Jean-Paul Leclercq** de m'avoir proposer un sujet aussi intéressant. Durant toute la durée de mes travaux, il a été disponible pour l'un ou l'autre problème que ce soit d'ordre "professionnel" ou relationnel. Je n'ai heureusement pas dû souvent y avoir recours, mais le fait de savoir qu'il était présent, a permis de décontracter l'atmosphère dès le début. Il m'a aussi souvent conseillé judicieusement sur les différentes orientations à donner à ce rapport.

Ensuite, je souhaite remercier Monsieur **Hubert Meurisse**. Il fût le contact direct au sein des Cliniques de Mont-Godinne et m'a fourni de nombreux conseils sur les orientations et les solutions déployées durant le stage. Je souhaite mettre en avant son écoute, sa disponibilité et son réel intérêt pour les résultats acquis.

Une troisième personne à remercier, et non des moindres, est **Cédric Peeters**. En effet, ma chance a été de rencontrer Cédric en qualité de chercheur au sein de l'Institut d'informatique. Il fût, pour moi, un grand support. Nos discussions ont permis d'éveiller en moi de nombreuses solutions ou pistes de solutions. Il est aussi un des principaux relecteurs de ce mémoire.

Je souhaiterais aussi remercier mes contacts du laboratoire de Mont-Godinne, en les personnes de Monsieur **Yvan Cornet** et du Docteur **Bernard Châtelain**. Bien que les rencontres avec ce dernier aient été peu fréquentes, il s'est toujours montré très intéressé par les travaux et a permis d'envisager d'autres visions pour le futur des recherches. **Yvan Cornet** a, pour sa part, été l'un des artisans de la réussite de ce stage. Bien que toujours fortement occupé, il a toujours su m'accorder quelques minutes de son temps pour répondre à mes questions. Il a aussi été en première ligne lors des tests des développements. Je tiens à le remercier pour sa disponibilité au jour le jour et pour ses nombreux coups de pouces dans l'usage du microscope.

Je remercie aussi **Raphaël Marée**, chercheur au sein du Département d'Électricité, Électronique et Informatique de l'ULg, pour sa disponibilité et son aide dans les recherches complémentaires et l'utilisation du logiciel "Pixit".

Á la suite de toutes ces personnes, je souhaite adresser mes remerciements les plus sincères aux personnes qui m'entouraient chaque jour. Á commencer par mes parents, relecteurs invétérés et véritables soutiens. Pour finir, les membres du secrétariat du laboratoire de Mont-Godinne qui ont bien voulu accueillir, dans leur paysage, un collègue inattendu.

Table des matières

Table des figures

Introduction

Les sciences de l'information sont de plus en plus présentes dans les activités humaines. L'évolution des télécommunications, des réseaux et des outils a entraîné le développement d'une multitude d'applications. Tous les secteurs sont touchés. Le secteur médical n'est pas épargné, on a notamment vu apparaître les notions de télémédecine et de télémicroscopie. Elles proviennent respectivement des activités de la médecine et de la microscopie auxquelles se sont greffées des pratiques de communication, de gestion et de partage de l'information.

Le stage réalisé au sein du laboratoire d'hématologie des Cliniques Universitaires de Mont-Godinne est à positionner dans la continuation des travaux de plusieurs stagiaires qui m'ont précédé. Ainsi, le travail de L. Zuyderhoff ([Zuy03]) portait sur la création de méga-images sur base d'images capturées de manière automatique par un microscope optique. Il s'agissait de mettre en place l'acquisition automatique et de réaliser la coregistration des images saisies.
Ensuite, G. Decock ([Dec04]) a réfléchi à l'utilisation du format JPEG2000 pour les méga-images ainsi qu'à une architecture client-serveur et caching afin "d'optimiser" l'utilisation des ressources disponibles, mémoire et processeur.
Par après, C. Jadoul ([Jad05]) s'est attelé à fournir une application de visualisation des images coregistrées. La solution imaginée est de type client-serveur et permet de consulter des frottis sanguins à distance (après la numérisation d'une partie de la plaquette sanguine). Dans le prolongement de ce développement, C. Peeters ([Pee07])a fourni une application permettant de localiser les globules blancs présents dans une méga-image.

Ce mémoire est réalisé dans le but de rappeler ces travaux, d'expliquer l'état de l'art en télémicroscopie au sein des Cliniques UCL de Mont-Godinne, de présenter les objectifs du stage, les choix qui ont été réalisés ainsi que les recherches et les avancées qu'elles ont amenées. Il s'articule en 4 parties :

La ***première partie*** permet au lecteur de prendre connaissance de l'état de l'art général en télémicroscopie, et plus particulièrement en microscopie virtuelle au laboratoire d'hématologie du Mont-Godinne. Je résumerai alors les travaux de mes prédécesseurs en reprenant les grands principes de leurs avancées. Pour clore cette première partie, j'aborderai les différents objectifs qui m'ont été proposés durant le stage[1]. Pour chacun d'eux,

[1] Au début du stage ou qui se sont imposés au fil des avancées sur les travaux

je dresserai un état de l'art afin de permettre au lecteur de connaître le point de départ de mon travail concernant chaque objectif.

La ***deuxième partie*** permettra de présenter les outils nécessaires à la compréhension des résultats obtenus au cours des recherches et avancées du stage.

La ***troisième partie*** dressera l'ensemble des résultats obtenus ainsi qu'une analyse de ceux-ci par rapport aux objectifs qui avaient été proposés. Elle reprendra donc les solutions imaginées pour résoudre le problème lié à l'autofocus. Ensuite, nous présenterons les coregistrations obtenues à partir d'images issues de sources différentes. Nous verrons que des erreurs sont apparues et quelle solution a permis d'y remédier. Enfin, une dernière partie portera sur les pistes imaginées pour la caractérisation et la classification des globules blancs, ainsi que l'architecture de l'application de visualisation développée par mes prédécesseurs.

La ***quatrième partie*** permettra de discuter les résultats et d'en préciser les limites. Cette section permet de mettre en relation ces résultats et les objectifs d'origine. Elle posera un regard critique sur les avancées et de proposer des pistes pour l'avenir des résultats obtenus.

La ***cinquième et dernière partie*** conclura ce mémoire en rappelant l'ensemble des objectifs et la manière dont ils ont été rencontrés.

Chapitre 1

Cadre et Objectifs

Introduction

Cette première étape consiste en la présentation du domaine sur lequel ont porté mes travaux. Dans cette optique, cette partie débutera en rappelant ce qu'est l'hématologie et l'intérêt des applications informatiques qui lui sont liées. Ensuite elle fournira une description de l'environnement de travail, c'est-à-dire le laboratoire d'hématologie des cliniques UCL de Mont-Godinne, et plus précisément les outils de travail : le microscope *"Olympus AX-70"* et le *"CellaVision DM96"*.

Ce cadre dressera ensuite un historique explicatif des travaux de mes prédécesseurs au sein du laboratoire d'hématologie. Car mon mémoire s'inscrit dans la continuité d'un projet de microscopie virtuelle ayant débuté depuis quelques années.

La dernière partie permettra d'expliciter les objectifs qui ont été fixés au départ du stage.

1.1 La télémédecine

Les technologies de l'information et de la communication ont intégré tous les secteurs et toutes les couches des organisations. La médecine n'y a pas échappé. Les TIC se sont immiscées dans tous les processus de la médecine. Dans les années 1990, profitant des avancées et de la convergence des technologies (bande passante, qualité d'image et de son,...), plusieurs développements ont été lancés avec pour objectif l'amélioration des soins de santé et de la condition des patients. La technologie permet d'agir directement sur le patient en apportant sa précision, ou indirectement, elle supporte alors les communications des experts relatives à la formation ou au diagnostic.

L'ensemble des applications de télémédecine peut être classifié sous forme de familles caractérisées par leur cadre d'utilisation. Ainsi, en 2003, un rapport sur l'état des lieux de la télémédecine en France [Haz03] dresse un ensemble de familles qu'il définit par :

télé-consultation ou télé-expertise : *échanger des avis entre professionnels de la santé (parfois multi-disciplinaire) ;*

télé-assistance : *assister à distance, principalement par des conseils diagnostiques et thérapeutiques, un patient localement démuni ;*

télé-surveillance : *surveiller à domicile, en ambulatoire, une fonction vitale défaillante ;*

télé-diagnostic ou télé-chirurgie : *pratiquer totalement et exclusivement à distance un acte médical ;*

cyber-réseau de santé : *organiser la circulation des données dans un réseau de santé ;*

cyberformation (ou e-learning) : *délivrer des informations, voire un enseignement ;*

cybermanagement (ou e-management) : *participer à la gestion des systèmes de santé ;*

e-santé : *famille élargie d'applications de cyber management qui vise à offrir aux patients un accès direct et permanent à leur dossier de santé ou à des télé-services médicaux ;*

Ce mémoire et ce stage ont abordé la télémédecine sous l'angle de la *télé-consultation, de la télé-expertise, du télé-diagnostic* et du *cyber-réseau de santé*, car le travail effectué a porté sur un système permettant de rendre disponible l'information médicale à plusieurs entités distantes. Chacune de ces entités peut alors donner un avis sur le diagnostic porté.

1.2 L'hématologie

L'hématologie est une branche de l'étude médicale ayant pour objet le sang. L'étude de celui-ci permet de valider un diagnostic. En effet, le sang contient différentes entités, chacune ayant un rôle bien déterminé tel que l'oxygénation, la coagulation et l'immunité. Le sang est constitué de 4 grandes classes d'éléments :

- ***les globules rouges*** (érythrocytes) : ils composent la plus importante part du contenu sanguin. Leur tâche est de transporter l'oxygène jusqu'aux cellules.
- ***les globules blancs*** (leucocytes) : ils interviennent dans la fonction immunitaire du corps. Il existe plusieurs types de globules blancs, chaque type ayant un rôle bien déterminé dans le processus de défense du corps.
- ***les plaquettes*** (trombocytes) : elles interviennent lors du processus de coagulation du sang. Ce processus est essentiel, il permet au sang de constituer les caillots. Ainsi, les plaquettes sont indispensables au processus de cicatrisation.
- ***le plasma*** : il est le composant liquide dans lequel les autres composants du sang sont baignés. Il contient en grosse majorité de l'eau (90%), mais aussi des substances provenant de la digestion (protéines, sels minéraux,...) ou des déchets du métabolisme (urée,...).

La figure 1.1, récupérée de [Pee07], permet d'illustrer chacun de ces composants.

Cet aperçu permet d'introduire l'importance de l'analyse des composants sanguins dans l'approche diagnostique. Lors de l'analyse de frottis sanguins, des défauts dans la

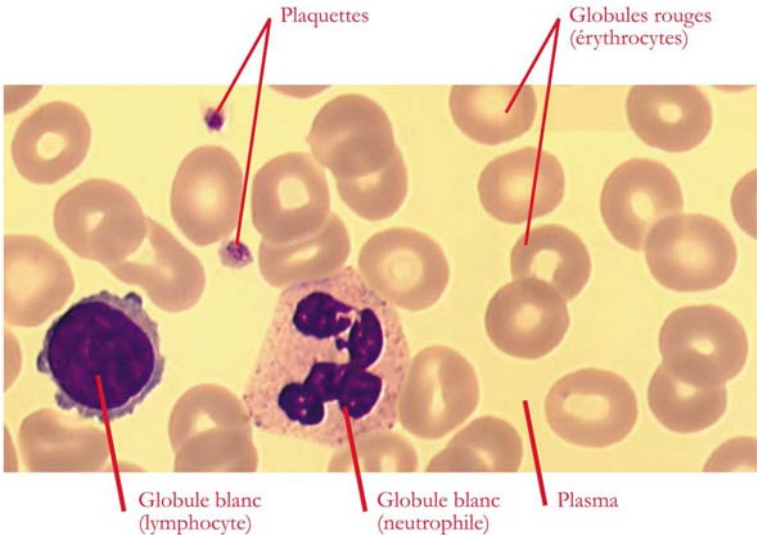

FIG. 1.1 – Frotti sanguin numérisé contenant les principaux composants du sang

constitution de ses composants ou plus simplement les rapports entre chacun d'eux sont des indices précieux dans le but de poser le diagnostic. Cela s'applique particulièrement aux globules blancs pour lesquels il existe plusieurs classes ayant une tâche bien définie dans le système immunitaire. Nous détaillerons les caractéristiques de chacune de ces classes au point 3.4.

1.3 Environnement de travail

Les travaux de mes prédécesseurs sont basés sur l'installation de microscopie disponible au cœur du laboratoire d'hématologie (voir figure 1.2).

Cette installation est le centre du développement d'une application de télé-microscopie. Ce terme rassemble les utilisations des technologies de l'information et de communication dans le domaine de la microscopie. La télé-microscopie a changé l'approche au microscope. Auparavant, l'expert posait une lame et l'observait pour prononcer un diagnostic, tout cela en manipulant directement le microscope. Les TIC ont ajouté une nouvelle dimension. Elles permettent de décentraliser la manipulation des lames mais aussi de mieux gérer les résultats d'analyses et offrent une plus grande base de connaissances en vue du diagnostic.

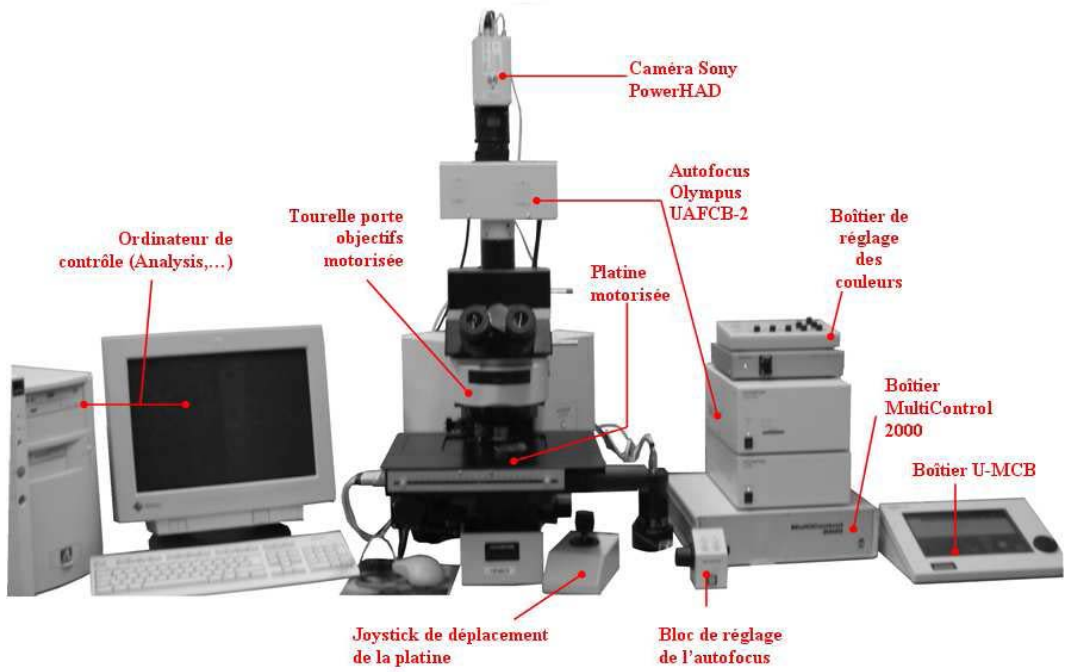

FIG. 1.2 – Environnement de travail autour du microscope

Deux grandes orientations de télé-microscopie ont été évoquées dans les travaux précédents [Geo02],[Pee07] :

l'approche dynamique, dite "real-time" : cette approche, consiste à décentraliser le contrôle du microscope. Celui-ci est manipulable à distance et en temps réel. Un inconvénient subsiste : la manipulation et le positionnement de la lame à observer.

l'approche statique, dite de "store and forward" : cette deuxième approche repose sur la communication et le stockage de l'information. Elle consiste à créer des images numériques des lames qui sont stockées auxquelles on peut accéder à distance et à n'importe quel moment. Les avantages sont que les copies numériques s'altèrent moins vite que les lames et qu'il est plus aisé d'obtenir différents avis sur le diagnostic posé. L'inconvénient principal est que l'observation d'une image offre moins de souplesse qu'une observation au microscope (cet inconvénient peut être contourné par plusieurs numérisations de la lame avec des objectifs différents).

La seconde vision a été retenue pour le laboratoire de Mont-Godinne au travers d'un microscope virtuel. Ce projet de microscopie virtuel, dont les évolutions sont décrites ci-après, tente de fournir les fonctionnalités du microscope tout en ajoutant les atouts d'un système distribué et d'un système informatique d'aide à la décision.

1.4 Évolution du projet au fil des années

Cette partie va permettre de reprendre les différentes étapes comprises dans ce qu'on appelle la microscopie virtuelle. Pour chacune de ces étapes, on trouvera un résumé des évolutions issues des travaux de B. Georges ([Geo02]), L. Zuyderhoff ([Zuy03]), G. Decock ([Dec04]), C. Jadoul ([Jad05]) et C. Peeters ([Pee07]). Cela permettra de dresser un historique du microscope virtuel de Mont-Godinne.

1.4.1 La capture d'une partie de lame

Cette étape permet d'obtenir une image numérique sur laquelle des applications en "store and forward" sont réalisables ou plus largement la détection automatique de caractéristiques de la lame observée.

La création d'une telle image nécessite un ensemble d'images de petites tailles qui, une fois mise côte à côte (coregistrées), formeront une méga-image. Par la suite, nous appellerons ces images, *"images sources"*. Au sein du laboratoire d'hématologie de Mont-Godinne, ces *images sources* sont capturées par une caméra Sony PowerHAD[1]. La taille des captures est de 768 pixels sur 573 pixels[2].

La saisie de ces images sources est commandée par le logiciel *"Analysis Pro 3.0"*[3]. *"Ce logiciel est conçu pour l'acquisition, l'analyse, le traitement et l'archivage d'images. Il présente plusieurs avantages. D'une part, il est doté d'un module comprenant des librairies permettant de gérer les différentes fonctionnalités du microscope AX70 d'Olympus, d'autres part, il possède un composant permettant à l'utilisateur de personnaliser l'application en créant lui-même ses modules personnels. Le langage utilisé par ce composant est un langage proche du ANSI-C appelé "Imagin-C" et qui a comme particularité de proposer un interpréteur intégré. Imagin-C est également compatible avec le "MS Windows Software Development Kit" (SDK) ce qui permet la programmation de nouvelles interfaces graphiques dans l'environnement "MS Windows". Le logiciel Analysis permet aussi une intégration facile de nouvelles fonctions dans son interface, en y associant des boutons.'*[Dec04]

Un module a été développé afin de permettre la capture automatique d'une série (galerie) d'images sources. L'utilisateur doit ajuster la caméra sur l'image de départ et ensuite paramètrer la taille de la zone à saisir (la taille de cette zone est décrite par le nombre d'images à capturer en largeur et en hauteur). D'autres paramètres, tel que l'identifiant de l'utilisateur ou encore le nom que portera la méga-image issue de ces images sources, sont aussi à définir de façon à identifier les captures sur l'espace de données.

Le module Analysis se charge de déplacer la platine supportant la lame pour modifier l'image sous le champ de la caméra.

Plus tard, le microscope a été équipé d'un autofocus qui ne se comporte pas bien dans certaines positions. Les erreurs générées font l'objet depuis quelques années de recherches de solutions et n'ont pas encore été résolues.

[1] Le laboratoire travaille aussi, depuis peu, avec une caméra Olympus DP71

[2] Cette résolution a été hard codée dans le module Analysis. Nous verrons que d'autres types de caméra gênèrent des images de résolutions différentes.

[3] voir `http://www.microscopy.olympus.eu/microscopes/Software_analysis_pro.htm`

Grâce à cet autofocus, la capture automatiquement produit des images nettes qui sont transmises à l'ordinateur de contrôle où elles sont stockées dans l'attente d'un traitement (dans notre cas, une coregistration).

1.4.2 La coregistration

Une fois les images sources disponibles, l'idée était de les utiliser pour construire une méga-image, une sorte de grand puzzle. Pour mener à bien cette idée, les travaux de L. Zuyderhoff ont permis de mettre au point une méthode de coregistration efficace[4].

Par la suite, une architecture client-serveur a été implémentée pour supporter l'activité de coregistration.

Cette architecture permet au serveur de commencer à coregistrer les images sources déjà disponibles pendant que Analysis continue la capture des images suivantes. Ainsi, lorsque Analysis effectue une capture, il utilise un client Java pour communiquer avec le serveur. Un langage de programmation fourni par Analysis[5], permet en effet de démarrer des programmes externes.

Ce client Java permet d'avertir le serveur de l'avancée de la coregistration. Il dispose de 2 opérations importantes :

1. l'opération *"NEW"* permet de demander au serveur de lancer une nouvelle coregistration. Celui-ci va démarrer la coregistration s'il trouve l'image source attendue, sinon il est mis en veille jusqu'à l'arrivée de cette dernière.
2. l'opération *"ADD"* permet d'annoncer au serveur la présence d'images sources additionnelles sur l'espace de données et ainsi le "réveiller" pour continuer la coregistration.

Le serveur distant, sachant où et quand il peut trouver les images sources, construit une méga-image dont la hauteur et la largeur (dont les valeurs exprimées sont en nombres d'images sources) auront été communiquées par le client.

1.4.3 Le stockage

Le serveur de coregistration conserve ses résultats en local. Une fois la mega-image construite, elle est stockée dans un dossier partagé du serveur. En fonction de l'utilisation de cette application, la taille des images pouvait rapidement devenir très importante.

A partir de cette observation, la réflexion s'est posée sur la façon de compresser et de stocker les méga-images. La solution mise en place principalement par L. Zuyderhoff et G. Decock (voir respectivement [Zuy03] et [Dec04]) est basée sur l'utilisation du standard de format de fichier JPEG2000. Ce format apporte des avantages multiples développés dans [Jad05]. Le premier avantage porte sur le stockage proprement dit. Il peut, en effet, compresser l'image avec ou sans perte. Ainsi, à taux de compression égal, une image JPEG2000 sera de meilleure qualité qu'une image JPEG.

[4]Nous verrons que celle-ci était efficace dans des conditions particulières, et qu'elle offrait peu de flexibilité.

[5]Pour plus d'informations, voir 3.2

Un autre avantage relevé est la possibilité d'agir sur des régions d'intérêts. Les parties d'une image comportant peu d'informations peuvent donc être encodées avec un taux de compression plus important et ainsi permettre un gain de place.

Dernier avantage utilisé dans l'application développée, la décompression peut être réalisée en différentes résolutions (figure 1.3). Cela se révèle très utile pour la visualisation.

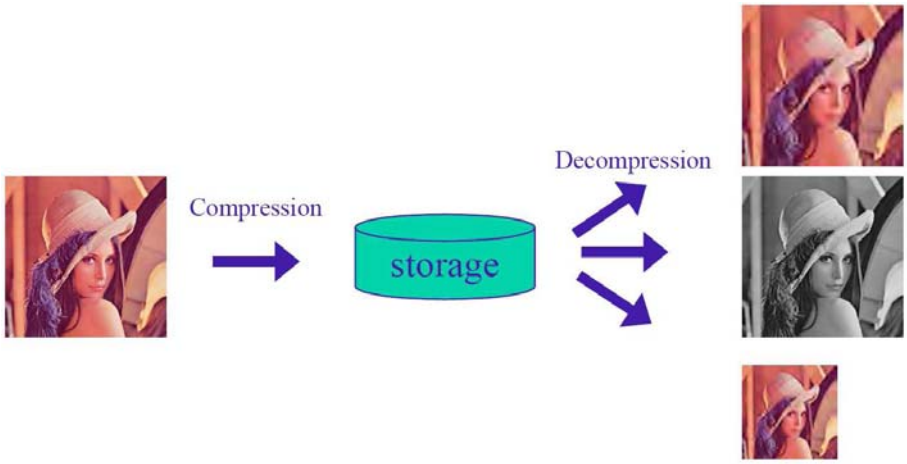

FIG. 1.3 – Différentes manières de décompresser une image JPEG2000 ([Des05])

D'autres avantages ont été mis en évidence tels qu'une meilleure robustesse face aux erreurs ou encore la possibilité d'une décompression progressive basée sur la qualité, la résolution ou l'espace (voir figure 1.4).

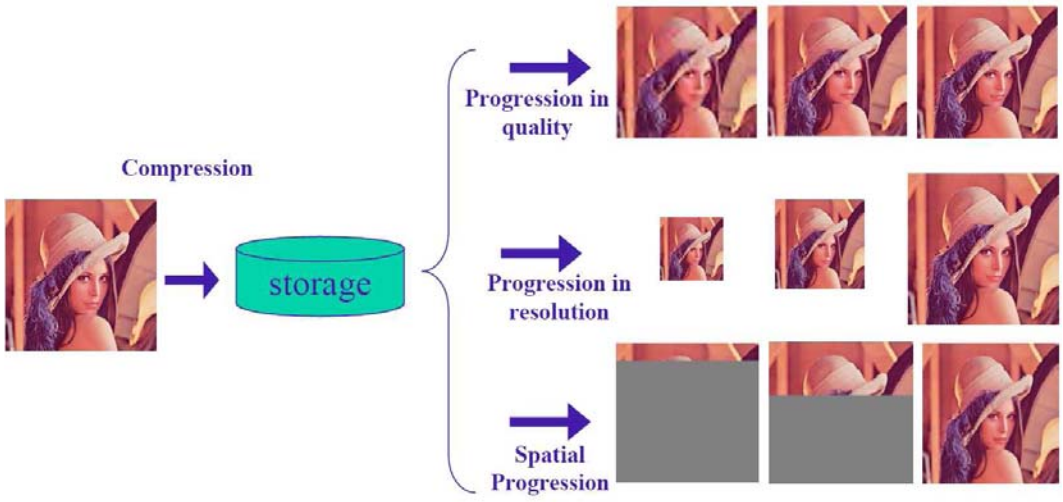

FIG. 1.4 – Différentes décompressions progressives ([Des05])

1.4.4 La visualisation

La troisième étape consiste en la visualisation des méga-images. L. Zuyderhoff a proposé une première solution basée sur une architecture client-serveur et sur un parallélisme avec un microscope traditionnel (cfr [Zuy03]).

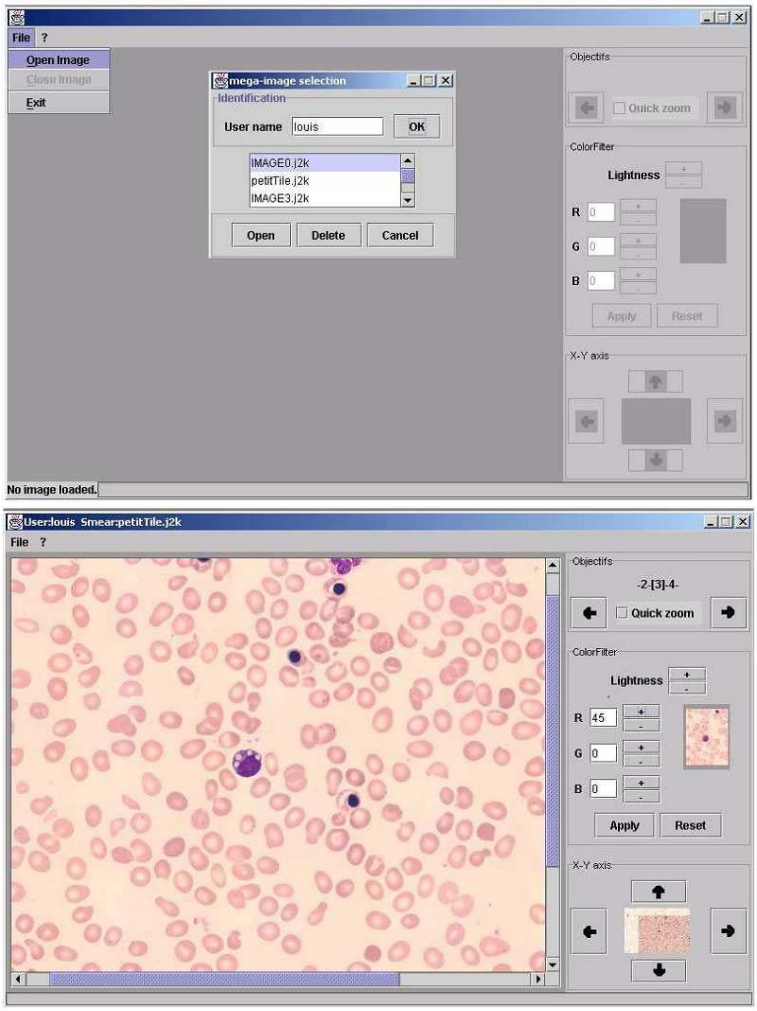

FIG. 1.5 – Interface du visualisateur développée par L. Zuyderhoff

Ainsi, des clients de visualisation soumettaient des requêtes de visualisation à un serveur de visualisation qui y répondait en fournissant les images décodées. La partie client proposait les opérations essentielles d'un microscope telles que les mouvements sur les axes X et Y, mais aussi un "zoom"[6]. Les déplacements sur les axes X et Y sont primordiaux afin de permettre à l'utilisateur de parcourir l'ensemble de la méga-image (échantillon nu-

[6]Ce zoom n'était en réalité qu'une fonctionnalité basée sur l'extrapolation de pixels. Si celle-ci devient trop importante, on observe une perte de qualité de la zone observée.

mérisé d'une lame). En effet, la taille des méga-images ne permet pas de les afficher lorsque le zoom est important. L'interface utilisateur proposait aussi des boutons permettant de modifier les caractéristiques de l'image telles que la luminosité ou des filtrages de couleurs.

Dans [Jad05], C. Jadoul posait 2 critiques sur l'application de visualisation développée. La première s'interrogeait sur la gestion de la mémoire. En effet, le client "visualisateur" conservait toutes les images décodées reçues du serveur (jusqu'à saturation). La deuxième critique portait sur les performances de l'application.

1.4.5 Le traitement

Après la visualisation, l'objectif était de réfléchir au traitement des méga-images dans le cadre d'une aide au diagnostic, ou simplement d'une meilleure visualisation. Cette opération, abordée par C. Peeters dans [Pee07] a été envisagée de deux manières. La première partie consistait en améliorant les méga-images par homogénéisation, la seconde en la localisation des globules blancs dans le frotti numérisé.

L'homogénéisation permet de réduire les variations de contraste dans les méga-images. En effet, les résultats de la coregistration comportaient des défauts issus des différences de luminosité lors de la capture des images sources. Lors de la coregistration, ces défauts étaient reportés sur le résultat. L'homogénéisation permet de réduire ces "disparités" pour une meilleure qualité des méga-images.

Cette étape est importante pour minimiser les erreurs lors de la seconde partie du traitement où la localisation des globules blancs et l'extraction de leurs caractéristiques sont effectuées. Ces opérations sont réalisées au moyen de méthodes de binarisation et de filtres d'images.

1.5 Transfert et visualisation de contenu d'archives ".mdb"

En parallèle au développement de ce microscope virtuel, le laboratoire s'est équipé d'un outil particulièrement efficace, ***CellaVision*** (figure 1.6). Il s'agit d'un logiciel qui permet d'analyser les échantillons sanguins envoyés au laboratoire. Il permet d'effectuer une classification automatique des globules contenus dans un frotti sanguin. Les résultats de ***CellaVision*** sont présentés sous forme de galerie d'images de globules blancs. L'utilisateur dispose de plusieurs fonctionnalités pour rectifier et ajuster la classification automatique.

Une fois l'analyse approuvée, les aperçus de globules blancs avec leurs descriptions sont conservés dans un premier temps dans une base de données propre à CellaVision. Le contenu des analyses est encore, à ce moment, consultable. Dans un deuxième temps, les laborantins peuvent alors valider l'analyse et en archiver les images et informations. L'archivage se réalise au moyen de base de données .mdb, de Microsoft Office Access. Ce format de fichier ne permet plus de consulter les résultats de l'analyse, ceux-ci deviennent inutilisables et inutiles.

Car, premièrement, l'ordinateur sur lequel fonctionne CellaVision ne dispose pas de logiciels permettant de lire des fichiers .mdb. Deuxièmement, même en supposant qu'un utilisateur fasse une copie pour lire le fichier .mdb sur une machine équipée, la lecture de ce fichier sera difficile au vu du nombre de tables qu'il contient[7].

FIG. 1.6 – CellaVision DM96

Jusqu'à présent, le laboratoire conservait les bases de données sur des disques durs externes. Ceux-ci se font de plus en plus nombreux, et représentent une masse de données (inutilisables) importante à gérer.

Il était donc intéressant de visualiser le contenu des .mdb (figure 1.7) mais aussi de la partager en stockant les informations utiles[8] sur des serveurs dédiés, partagés. L'accès à ces serveurs est possible à partir d'autres stations de visualisation que celles du laboratoire ce qui permet d'obtenir d'autres avis médicaux.

L'idée a été mise en place dans les travaux de Cédric Peeters. Son application permettait déjà d'extraire des informations de fichiers .mdb (Microsoft Office Access) et de formater une bonne partie des données de la base de données en des objets DICOM pouvant être envoyés sur le serveur distant.

[7]Il contient 73 tables. Plus le nombre d'analyses que le fichier contient est élevé, plus la lecture est difficile.

[8]Les informations utiles sont des informations comme les images des globules détectés, les informations de classification donnés par l'analyse CellaVision mais aussi les informations concernant le patient

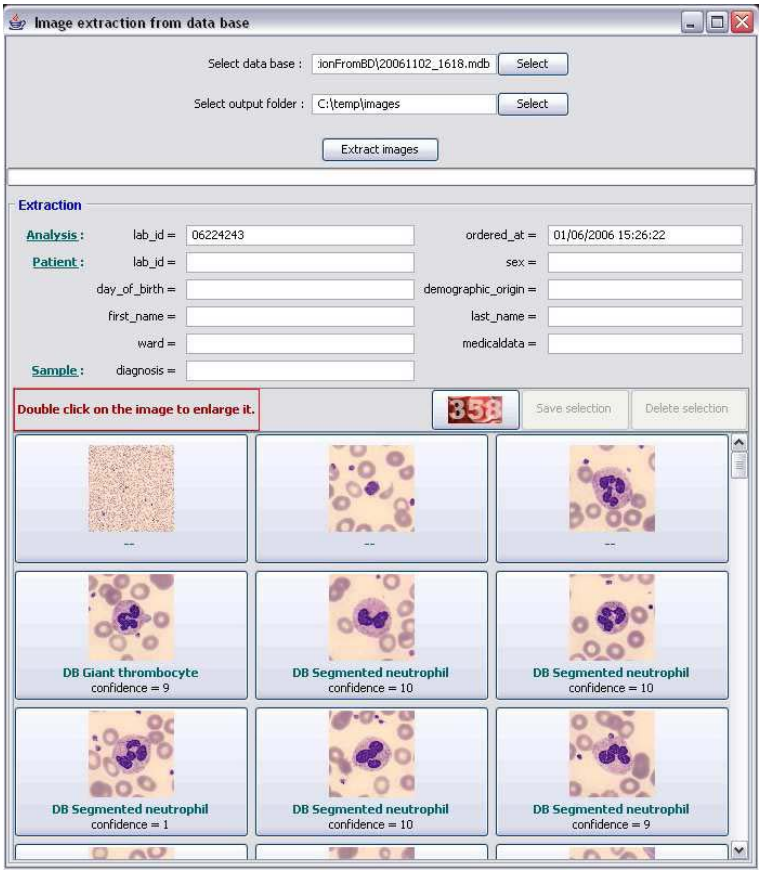

FIG. 1.7 – Interface d'extraction développée par Cédric Peeters

1.6 Objectifs du mémoire

Les objectifs du stage et du mémoire ont été multiples. Il était bien entendu question de prendre place dans une organisation et d'apporter ma pierre à cet édifice.

Plusieurs objectifs m'ont été présentés, tous portant sur des domaines très différents les uns des autres. Mon travail consistait donc en la recherche, l'amélioration, l'approfondissement ou encore la correction des solutions logicielles précédemment développées.

L'autofocus : Le premier objectif défini était de solutionner les problèmes liés à l'autofocus du microscope *Olympus AX-70*. Ce problème intervient lors de l'étape de capture de la lame (voir 1.4.1) et a déjà été soulevé par L. Zuyderhoff (dans [Zuy03]). B. Permentier, stagiaire issu de l'IESN[9], (cfr [Per06]), et C. Peeters (cfr [Pee07]) ont tous deux exploré diverses solutions, mais celles-ci n'ont pas donné de résultats satisfaisants ou ont dû être abandonnées suite à des difficultés d'ordre matériel. Chacun de ces travaux avait permis

[9]Institut d'Enseignement Supérieur de Namur devenu Haute École de Namur (HENam) par fusion avec Haute école namuroise catholique (HENaC)

de mettre en évidence le problème, de fournir et d'écarter des pistes de solutions.

Pour rappel, le microscope est muni d'une caméra permettant de faire des captures d'une partie de lame. Afin de pouvoir traiter plus rapidement de manière digitale les images capturées, un autofocus a été monté sur le microscope. Il s'agit d'un autofocus hardware (et non logiciel). Il permet donc d'obtenir "automatiquement" des images nettes, il n'est plus nécessaire que l'utilisateur soit présent à chaque capture pour faire le point. L'avantage qui en découle, est un gain de temps très important.

Il reste que cet autofocus ne fonctionne pas bien. Dans certains cas limites, l'autofocus ne trouve pas de point d'équilibre et renvoie une erreur à l'application.

En effet, l'autofocus ne fait le point que sur une petite partie de la partie visible. Il arrive donc que la mise au point ne puisse pas se réaliser de manière automatique car la zone de traitement n'est pas assez large et ne contient pas assez de contraste pour mener à bien le focus. Cette erreur n'était à ce moment là pas encore gérée. Cet objectif consistait donc en l'exploration de différentes solutions pour rendre l'application robuste face aux erreurs de l'autofocus.

Extraction des informations : Il m'a aussi été demandé de compléter l'application développée par Cédric Peeters en 2006-2007. Cette application offrait aux laborantins un service d'archivage partagé et de visualisation des résultats d'analyses des patients.

Comme expliqué au point 1.5, une fois archivée, les informations ne sont plus visualisables par les laborantins. C'est, pour remédier à cette perte d'informations, que Cédric Peeters a travaillé au développement de cette application d'extraction d'informations de la base de données Access. L'application a été développée dans deux optiques :

- La première était de fournir un moyen de consulter l'analyse même lorsque celle-ci a été archivée. Une interface a donc été créée pour faire apparaître son contenu.
- La deuxième vision était de rendre ce même contenu utilisable, visualisable, par un plus grand nombre d'experts. Il ne fallait pas se limiter à une visualisation locale. Pour cela, l'application pouvait être démarrée sous forme de service. Ce service permettait de créer des objets DICOM sur base des informations contenues dans la base de données. Ces objets étaient alors envoyés sur un serveur partagé.

Le choix entre le démarrage de l'application orienté service ou interface se réalisait en fonction du nombre d'arguments passés au démarrage.

Cette application n'avait pas pu être finalisée par manque d'informations. En effet, le logiciel "CellaVision" créait des champs permettant d'identifier et de documenter le patient mais ces champs n'étaient malheureusement pas remplis. Dans l'attente d'un update de "CellaVision", des valeurs de tests avaient été assignées.

Suite à la mise à jour de CellaVision, mon objectif était de compléter les champs DICOM adéquats à l'aide des informations qui pouvaient, à présent, être récupérées.

Caractérisation et classification de globules blancs : Le troisième objectif, introduit par les travaux de Cédric Peeters (cfr [Pee07]) propose de définir un ensemble de caractéristiques permettant de dériver des règles de classification de globules blancs. Comme décrits au point 1.4.5, les précédents développements avaient été orientés vers l'amélioration des images, avant le traitement proprement dit. Ainsi, après avoir homogénéiser la luminosité des méga-images, Cédric Peeters fournit une solution intégrant la localisation des globules blancs dans les méga-images.

Dans la suite de la localisation , il est évident qu'il est intéressant de traiter individuellement chacune des cellules repérées. Dans cette optique, l'application déjà développée peut être complétée d'un module de traitement de ces règles permettant d'associer, à chaque globule blanc détecté, une classification. Il est maintenant nécessaire de recenser les différents types de composant du sang et de les décrire au travers des critères mathématiques utilisables par l'ordinateur.

Coregistration d'images de grande résolution : Un quatrième objectif est apparu au début du stage. En effet, au moment de prendre connaissance du fonctionnement des développements précédents, il est apparu que le laboratoire équipait le microscope optique AX-70 d'une autre caméra pour capturer des images plus larges. Cette caméra est numérique et fonctionne avec un autre logiciel d'acquisition. Comme aucun développement n'a encore été réalisé pour ce nouveau logiciel, les utilisateurs capturent les images successives "à la main", c'est-à-dire qu'ils déplaçaient manuellement la platine du microscope et, une fois celle-ci correctement positionnée, lançaient la capture du champ visualisé. Il est bien évident que cette manoeuvre nécessite beaucoup de temps et de précision.

Cette étape de capture n'a, pour le moment, pas fait l'objet de recherches ou de développements. Plusieurs questions ont ainsi été posées, telles que :

- serait-il possible d'utiliser la nouvelle caméra avec le logiciel Analysis ?
- serait-il possible de concevoir des modules équivalents à ceux développés précédemment pour Analysis ? Le nouveau logiciel offre-t-il la possibilité de les implémenter en interne ?
- serait-il possible d'utiliser le serveur de coregistration mis en place dans le passé avec le nouveau format d'images obtenu ?

Les deux premières questions ont été écartées. En effet, pour la première question, il pourrait être possible d'utiliser Analysis, mais il faudrait se procurer les pilotes adéquats. En ce qui concerne la seconde question, elle n'a été que citée, aucune réflexion n'a suivi.

La troisième question a donc été conservée. Ainsi est apparu le dernier objectif : tester le serveur de coregistration créé par Louis Zuyderhoff avec les nouvelles images sources et si les résultats ne sont pas satisfaisants, modifier le serveur pour le rendre indépendant du format et surtout de la résolution des images à coregistrer.

L'ensemble de ces objectifs se positionnent dans une suite de développements précédents et visent à améliorer et compléter :

- l'autofocus est un problème qui a déjà été abordé par plusieurs de mes prédécesseurs sans pour autant apporter une solution. Une nouvelle piste est apparue. Elle sera explorée et détaillée dans ce mémoire.
- l'extraction d'informations de base de données est un développement débuté en 2007 par Cédric Peeters. Partant de ça, j'ai dû compléter l'application après quelques mises-à-jour sur le logiciel CellaVision. Mais nous verrons que l'application a dû être modifiée en profondeur car elle ne permettait pas de répondre entièrement aux attentes.
- la caractérisation des globules blancs est une perspective déjà présentée depuis quelques années. Dans cette optique, mes travaux ont porté sur l'extraction de caractéristiques supplémentaires, la découverte des types des globules blancs et les critères utiles à leur classification.
- la coregistration d'images est un principe déjà traité par L. Zuyderhoff (dans [Zuy03]) et G. Decock (dans [Dec04]) mais qui nécessite des améliorations pour accepter différentes résolutions d'images (et dans une moindre mesure, de format de fichiers images).

Mes travaux ont donc bénéficié d'une multitude de sources et de connaissances, me permettant de démarrer sur de bonnes bases.

Dans la suite de ce mémoire, nous reprendrons ces objectifs en décrivant pour chacun d'eux les outils utilisés, les résultats obtenus et les propositions de pistes à suivre pour la suite des travaux.

Chapitre 2

Matériel et Méthodes

2.1 Introduction

Cette partie intitulée "Matériel et Méthodes" permet de fournir au lecteur un ensemble de données utiles à la compréhension des résultats.

Elle décrira les outils et les techniques utilisées dans les développements de chacun des objectifs.

Pour la coregistration d'images, nous rappellerons le fonctionnement de la méthode de coregistration mise en place par L. Zuyderhoff. Il est en effet important de la comprendre pour voir les problèmes qui peuvent apparaître.

Pour la caractérisation de globules blancs, nous décrirons, premièrement, les différents composants du sang. Nous reprendrons aussi un arbre de classification utilisé par les laborantins. Cet arbre nous permettra de mettre en évidence des caractéristiques utilisables mathématiquement et automatiquement dans un module logiciel. En effet, à partir de ces caractéristiques, il pourrait être possible de mettre en place un système de règles.

Le deuxième point abordé portera sur un traitement de l'image numérique. Nous présenterons différentes méthodes d'extraction de contours. Les contours sont, nous le verrons, un élément important à caractériser.

Pour l'objectif de visualisation et d'envoi du contenu d'archives *Access*, nous rappellerons simplement le fonctionnement de l'accès aux bases de données ainsi qu'un bref rappel du standard DICOM et de son utilisation au sein des Cliniques Universitaires de Mont-Godinne.

Pour l'objectif portant sur l'autofocus, nous décrirons la communication RS232 utilisée entre l'ordinateur de contrôle et le microscope.

2.2 Le langage Java

Le premier choix se posait sur le langage de programmation à utiliser pour les développements. Le langage Java s'est imposé de lui-même. Le choix a été réalisé par mes prédécesseurs et a été conservé pour les nouveaux développements.

Java est un langage orienté objet, apparu récemment en 1996.

Au départ, Java a été conçu en suivant 5 objectifs [1] :

simple, orienté-objet et familier : simple, il ne nécessite pas de formation intensive et peut être rapidement appris. Il est orienté-objet, ce qui permet de concevoir des applications distribuées basées sur les réseaux. Il offre aussi des librairies éprouvées qui fournissent des interfaces pour les entrées et sorties ou le réseau mais aussi pour les interfaces utilisateur. Enfin, familier car il a été construit en se basant sur C++ tout en supprimant les complexités de ce langage.

robuste et sûr : Java est un langage hautement fiable. Il offre deux niveaux de vérification : à la compilation et aussi à l'exécution. Le programme ne gère pas les pointeurs de données, tout est géré en interne (appel au garbage collector) ce qui permet d'éviter les erreurs qui peuvent y être liées. Sur le plan de la sécurité, la technologie Java dispose de dispositifs de sécurité qui empêchent l'invasion par l'extérieur.

portable et d'architecture neutre : la technologie Java aspire à être déployée indifféremment de l'environnement. Le caractère interprété de Java permet de solutionner les problèmes de distribution et de version. Ainsi, la Java Virtual Machine (JVM) permet d'apporter la portabilité aux applications développées en Java.

haute performance : la performance est importante et le garbage collector automatique y participe. Défini comme un processus interne de basse priorité, il permet de disposer de mémoire suffisante quand c'est nécessaire, améliorant au passage la performance. Il offre aussi des interfaces donnant la possibilité d'écrire des applications nécessitant de grandes ressources de calcul en code machine.

interprété, "threadé", et dynamique : le caractère interprété de Java permet d'accélérer les cycles de développements, et de développer des applications plus légères. Java permet aussi le multi-threading grâce à des primitives de synchronisation de haut niveau ce qui permet plus d'interactivité avec les utilisateurs. Enfin, il est dynamique à l'exécution car les classes sont utilisées seulement en cas de besoin.

L'ensemble de ces caractéristiques rejoint les objectifs du système à concevoir. Il était nécessaire de pouvoir facilement mettre en place un système distribué, performant, portable et permettant la création de modules au fil des années. De plus, durant la formation de maître en informatique, plusieurs projets ont été développés en Java, ce qui m'a permis d'acquérir de l'expérience et un goût certain pour ce langage de programmation.

[1] extrait de `http://java.sun.com/docs/white/langenv/Intro.doc2.html`

2.3 Méthode de coregistration d'images

Cette section va permettre d'expliquer la méthode de coregistration implémentée par Louis Zuyderhoff. La coregistration est une étape importante dans la création des méga-images au sein du laboratoire d'hématologie de Mont-Godinne.

Cette méthode de coregistration a été définie dans le mémoire de L. Zuyderhoff ([Zuy03]) et reprise par G. Decock ([Dec04]) comme étant *"le processus d'alignement ou de mise en correspondance de deux images, appelées "image maître" et "image esclave" recouvertes entièrement ou partiellement par une même zone. Cette zone sous-entend donc que les images représentent entièrement ou partiellement une même information que l'on appellera la "scène observée""*. Cependant, cette définition a été énoncée de manière générale. Le système proposé à la suite de celle-ci est moins souple. En effet, la définition met l'accent sur un recouvrement partiel ou complet des images. Or le système, nous le verrons, ne fonctionne que sur des zones de recouvrement déterminées et bornées arbitrairement.

Prenons un peu de temps pour rappeler le fonctionnement de la coregistration proposée. Elle est décomposée en quatre grandes étapes :

- **une première étape appelée *"de sélection du vecteur étalon"* (dans l'image maître) :** cette première étape va permettre de détecter dans l'image maître un vecteur de pixels aligné qui offre le plus de contraste. Ce meilleur contraste va permettre d'obtenir de meilleurs résultats dans les comparaisons entre les pixels de l'image maître et ceux de l'image esclave.
- **une deuxième étape dite *"de sélection de la zone de recherche"* (dans l'image esclave) :** cette seconde étape consiste en un calcul de la position du rectangle contenant les pixels de l'image esclave qui seront les premiers pixels des vecteurs à comparer avec le vecteur étalon.
- **une troisième étape *"de comparaison"* :** lors de cette étape est effectuée la comparaison entre le vecteur étalon (issu de l'image maître) et les vecteurs à comparer (issus de l'image esclave). Le résultat attendu est la position dans l'image esclave du vecteur à comparer le plus semblable au vecteur étalon.
- **une quatrième et dernière étape nommée *"de positionnement relatif"* (par rapport à l'image maître) :** cette étape va clore le processus de coregistration en calculant le décalage de l'image esclave par rapport à l'image maître sur base des décalages issus de l'opération de comparaison.

Pour les développements suivants, nous considérerons les notations suivantes[2] :

Soit
- M_{ij}, la valeur du pixel de l'image M aux coordonnées (i,j)
- M_i la colonne à l'index i
- l la largeur de l'image
- h la hauteur de l'image
- ERROR_MAX_VERT une "erreur" verticale passée via le fichier de paramètres
- ERROR_MAX_HORI une "erreur" horizontale passée via le fichier de paramètres
- $BufV$ un tableau vertical de pixels noirs
- v_{buff} la longueur de $BufV = (l - (2 * 40))$
- $BufH$ un tableau horizontal de pixels noirs
- h_{buff} la longueur de $BufH = (h - (2 * 40))$

Afin de donner un aperçu de l'ordre de grandeur, dans les coregistrations d'images de 768 sur 573 pixels, les valeurs de ERROR_MAX_VERT et de ERROR_MAX_HORI sont fixées à 30 pixels.

2.3.1 L'étape de sélection du vecteur étalon

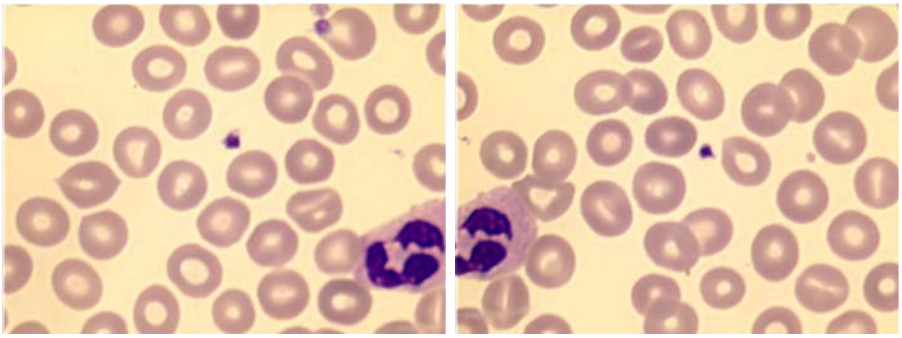

FIG. 2.1 – Deux images à coregistrer, l'image maître (à gauche) et l'image esclave (à droite)

Lors de la première étape, l'algorithme détermine un tableau de pixels ayant le plus d'informations potentiellement utiles pour effectuer les comparaisons. La figure 2.1 montre la situation, l'image esclave doit être "unie" à l'image maître. Ainsi, de l'image maître, on extrait une colonne (ou ligne) de pixels la plus "contrastée", appelée **vecteur étalon**.

Ce choix se déroule en plusieurs étapes :
En considérant une **coregistration horizontale**,

1. On détermine une zone dans laquelle sera choisi ce vecteur étalon. Cette zone de sélection est délimitée par un "rectangle[3]" **horizontal**
   - de hauteur $h_{ms} = 1$ pixel ;
   - de largeur $l_{ms} = (100 - (2 \times \text{ERROR\_MAX\_VERT}) + 5)$ pixels ;

[2] Les indices des matrices sont inversés pour coller à l'utilisation des tableaux en code Java
[3] Ce rectangle est en réalité un vecteur.

- positionné à une distante $y_{ms} = 40$ pixels du bord supérieur ;
- et une distance x_{ms} du bord droit = $(l - 100 - \text{ERROR\_MAX\_HORI} + 5)$.

2. On prélève, dans l'image master, l_{ms} colonnes t_i de taille v_{buff} pour $(x_{ms},\ y_{ms})$ jusqu'à $(x_{ms},\ y_{ms} + l_{ms})$. L'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{v_{buff-1}} t_{ij}^2} \qquad \forall\ \ 0 \leq i < l_{ms}$$

et il calcule aussi la norme de $BufV$ initialisé par des pixels noirs, notée $\|BufV\|$[4] ;

3. Un indice de comparaison c_i relatif à chaque colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{v_{buff-1}} |BufV_j - t_{ij} \times \frac{\|BufV\|}{\|t_i\|}| \qquad \forall\ \ 0 \leq i < l_{ms}$$

4. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{l_{ms}-1})$$

Soit ve, le tableau de pixels de référence correspondant au minimum des indices. Il est appelé vecteur étalon et la coordonnée de son premier pixel est $(x_{ve},\ y_{ve})$[5]

Si la **coregistration est verticale**, le mécanisme est le même, seules les bornes sont modifiées de la manière suivante :

1. On détermine une zone dans laquelle sera choisi ce vecteur étalon. Cette zone de sélection est délimitée par un "rectangle[6]" **vertical**
   - de hauteur $h_{ms} = (100 - (2 \times \text{ERROR\_MAX\_HORI}) + 5)$ pixels ;
   - de largeur $l_{ms} = 1$ pixel ;
   - à une distance $y_{ms} = (h - 100 - \text{ERROR\_MAX\_VERT} + 5)$ pixels du bord supérieur ;
   - et à une distance $x_{ms} = 40$ pixels du bord droit .
2. On prélève, dans l'image master, h_{ms} lignes t_i de taille h_{buff} pour $(x_{ms},\ y_{ms})$ jusqu'à $(x_{ms},\ y_{ms} + h_{ms})$. L'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{h_{buff-1}} t_{ij}^2} \qquad \forall\ \ 0 \leq i < h_{ms}$$

et il calcule aussi la norme du vecteur $BufH$ initialisé par des pixels noirs, notée $\|BufH\|$[4] ;

[4] Concrètement, $\|BufV\| = \|BufH\| = 0$ vu que ces buffers sont composés de pixels noirs, ce qui implique que $\forall\ \ i,\quad c_i = 0$ et donc, dans tous les cas, le dernier vecteur analysé sera choisi comme vecteur étalon. Nous montrerons plus tard comment remédier à ce problème.

[5] Donc, dans le cas horizontal, $y_{ve} = y_{ms}$, et $x_{ms} \leq x_{ve} < x_{ms} + l_{ms}$.

[6] ce rectangle est en réalité un vecteur.

3. Un indice de comparaison c_i relatif à chaque colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{h_{buff-1}} |BufH_j - t_{ij} \times \frac{\|BufH\|}{\|t_i\|}| \qquad \forall \ \ 0 \leq i < h_{ms}$$

4. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{h_{ms}-1})$$

Soit ve, le tableau de pixels de référence correspondant au minimum des indices. Il est appelé vecteur étalon et la coordonnée de son premier pixel est (x_{ve}, y_{ve})[7]

Illustration : Pour illustrer le développement ci-dessus, reprenons l'image maître de la figure 2.1 et observons quelle zone est désignée. Précisons que les images ont une taille de 768 sur 573, et que le fichier de paramètre de coregistration fixe la valeur de ERROR_MAX_VERT et ERROR_MAX_HORI à 30 :

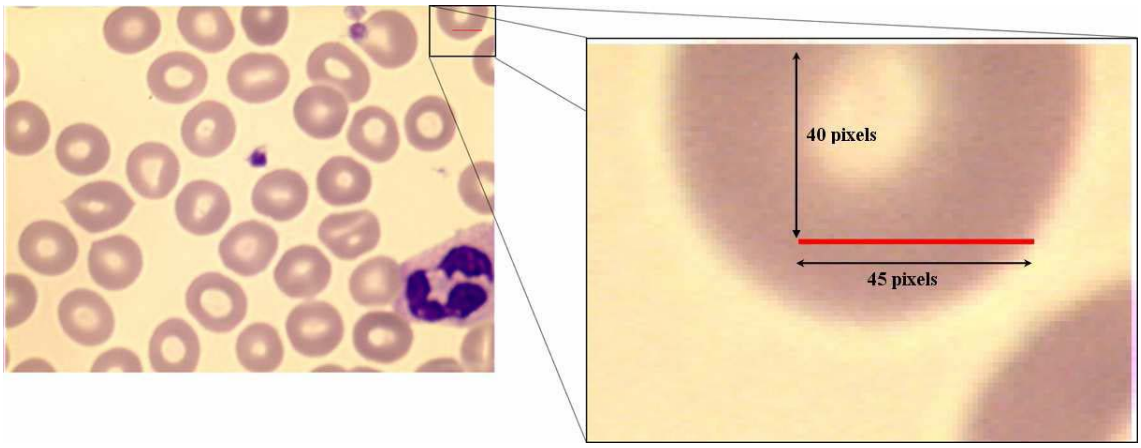

FIG. 2.2 – Détection du tableau de pixel témoin, vecteur étalon, dans l'image maître

Pour chacun des pixels rouges, on extrait une colonne de taille 493 que l'on compare au tableau de 493 pixels noirs. La valeur 493 étant donnée par v_{buff} c'est-à-dire $573 - (2 \times 40)$, 40 étant fixé arbitrairement dans le programme. La colonne choisie est celle dont les coordonnées du premier pixel est $(733, 40)$

[7]Dans le cas vertical, $x_{ve} = x_{ms}$, et $y_{ms} \leq y_{ve} < y_{ms} + h_{ms}$

2.3.2 L'étape de sélection de la zone de recherche

La seconde opération consiste en la sélection de la zone de recherche dans l'image esclave. Cette zone déterminera l'ensemble des pixels à partir desquels seront prélevées les colonnes (lignes) si la coregistration est horizontale (verticale).

La zone de sélection est délimitée par l'algorithme de la manière suivante :

Dans le cas horizontal,

- La position du coin supérieur gauche, noté (x_{slave}, y_{slave}) est donnée par :
  - $x_{slave} = x_{ve} - (x_{ms} - 5)$;
  - $y_{slave} = 5$;
- La largeur de cette zone, $l_{slave} = (2 * \text{ERROR\_MAX\_VERT})$;
- La hauteur de cette zone, $h_{slave} = h - v_{buff}$;

Dans le cas vertical,

- La position du coin supérieur gauche, noté (x_{slave}, y_{slave}) est donnée par :
  - $x_{slave} = 5$;
  - $y_{slave} = y_{ve} - (y_{ms} - 5))$;
- La largeur de cette zone, $l_{slave} = l - h_{buff}$;
- La hauteur de cette zone, $h_{slave} = (2 * \text{ERROR\_MAX\_HORI})$;

Illustration : Pour illustrer la sélection de la zone de recherche, observons le résultat obtenu sur la figure 2.3 et reprenons l'image esclave (de résolution identique à l'image maître c'est-à-dire 768 sur 573) de la figure 2.1 et observons quelle zone est désignée. Le fichier de paramètre de coregistration n'a pas changé.

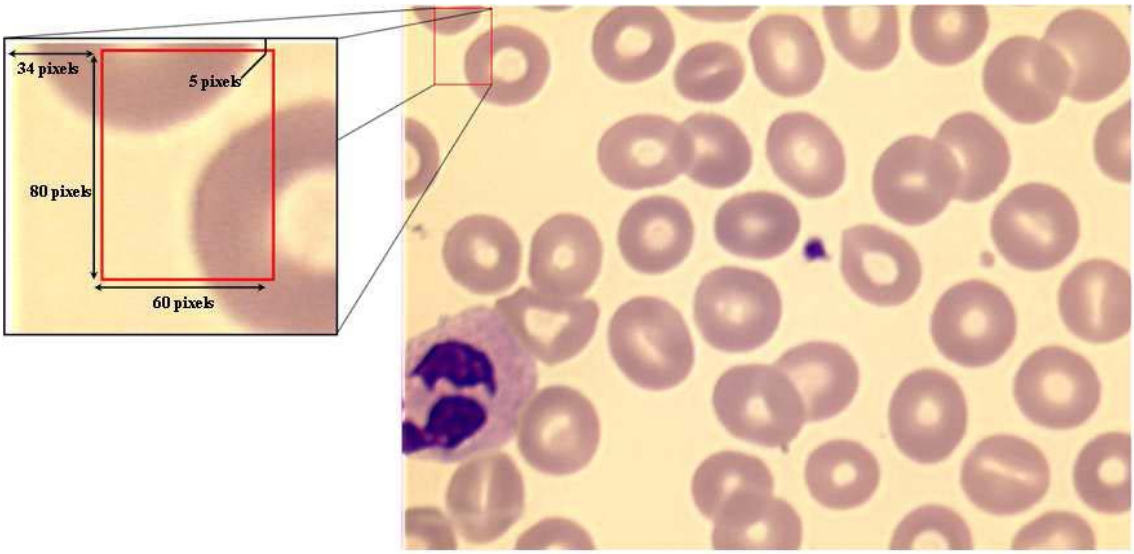

FIG. 2.3 – Sélection de la zone de recherche dans l'image esclave.

2.3.3 L'étape de coregistration

La troisième étape consiste en la comparaison du vecteur étalon trouvé au point 2.3.1 avec les vecteurs construits au départ des pixels de la zone de recherche déterminée au point 2.3.2. Pour chacun des pixels de la zone de recherche, une colonne (ou ligne) est extraite. Cette colonne (ou ligne), de longueur v_{buff} (ou b_{buff}) est comparée pixel par pixel au vecteur étalon de même taille.

L'algorithme utilise la même méthode que celle présentée au point 2.3.1 à une différence près : le buffer de pixels noirs est remplacé par le vecteur étalon déterminé à cette même étape.

On obtient donc :
Dans le cas horizontal,

1. Pour les $l_{slave} * h_{slave}$ colonnes t_i de taille v_{buff} de cette zone, l'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{v_{buff}-1} t_{ij}^2} \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

   et il calcule aussi la norme du vecteur étalon ve, notée $\|ve\|$;

2. un indice de comparaison c_i relatif à la colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{v_{buff}-1} |ve_j - t_{ij} \times \frac{\|ve\|}{\|t_i\|}| \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

3. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{l_{slave} * h_{slave}-1})$$

   Soit v_{sol}, le tableau de pixels dont les pixels sont les plus semblables à ceux du vecteur étalon. Ce tableau commence au pixel de coordonnées (x_{sol}, y_{sol}) et a, pour rappel, une longueur de v_{buff}.

Et pour une coregistration verticale, la méthode est la même mais on parle de lignes et non de colonnes (ce qui implique que la longueur des tableaux de pixels manipulés est h_{buff} et non v_{buff}) :

1. Pour les $l_{slave} * h_{slave}$ lignes t_i de taille h_{buff} de cette zone, l'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{h_{buff}-1} t_{ij}^2} \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

   et il calcule aussi la norme du vecteur étalon ve, notée $\|ve\|$;

2. un indice de comparaison c_i relatif à la colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{h_{buff}-1} |ve_j - t_{ij} \times \frac{\|ve\|}{\|t_i\|}| \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

3. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{l_{slave} * h_{slave}-1})$$

Soit v_{sol}, le tableau de pixels dont les pixels sont les plus proches de ceux du vecteur étalon. Ce tableau commence au pixel de coordonnées (x_{sol}, y_{sol}) et a, pour rappel, une longueur de h_{buff}.

Illustration : L'illustration 2.4 suivante montre qu'après avoir choisi le vecteur étalon (en vert dans l'image maître), on prélève les colonnes pour chacun des pixels du rectangle noir dans l'image esclave. Pour chaque pair vecteur étalon - vecteur à comparer, on obtient un indice de comparaison. Une fois, que l'indice a été calculé pour chacun des vecteurs à comparer (en rouge), celui qui a l'indice de comparaison minimum est désigné comme le plus semblable au vecteur étalon (vecteur vert dans l'image esclave).

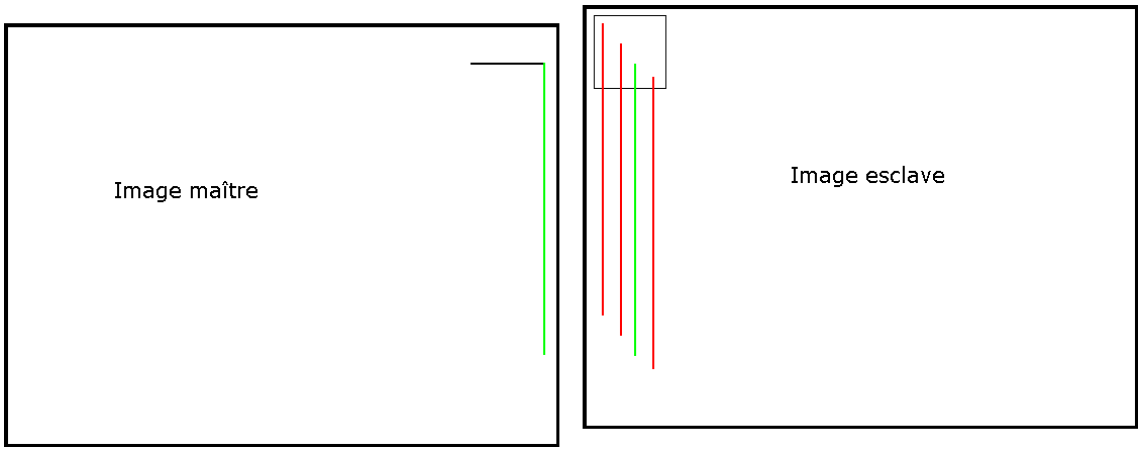

FIG. 2.4 – Coregistration de deux images

2.3.4 L'étape de positionnement relatif

Cette dernière étape n'est pas aussi importante que les précédentes et vise à calculer la valeur de décalage du pixel de coordonnée (0,0) dans l'image maître. Le résultat obtenu lors de l'opération précédente n'était que la valeur de la colonne (ligne) qui correspond le plus au vecteur étalon. Il faut à présent calculer le décalage à imposer à l'image esclave à partir de la position de cette colonne et de la position du vecteur étalon.

Pour cela, l'algorithme utilise les opérations suivantes :

- pour le cas horizontal :
  - $x = x_{slave} + \text{ERROR\_MAX\_HORI} - x_{sol}$
  - $y = y_{ve} - y_{sol}$;
- pour le cas vertical :
  - $x = x_{ve} - x_{sol}$
  - $y = y_{slave} + \text{ERROR\_MAX\_VERT} - y_{sol}$;

Illustration : Pour illustrer cette étape de positionnement, reprenons le cas des illustrations précédentes. Rappelons que les images ont une résolution de 768 pixels sur 573. La figure 2.5 permet de visualiser le positionnement des différents points survenant dans la coregistration.

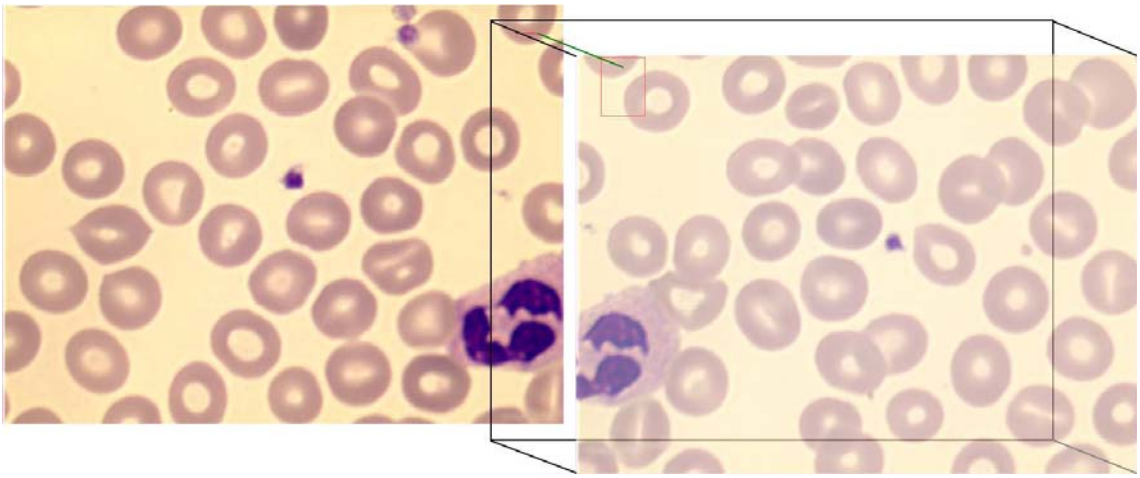

FIG. 2.5 – Positionnement relatif de l'image esclave par rapport à l'image maître

2.4 Caractéristiques d'images numériques

2.4.1 Détection de contours

La détection de contours est un processus important dans le traitement d'images. Une image numérique peut être considérée comme un ensemble de pixels caractérisé par une couleur (en niveau de gris ou non). On peut aussi considérer une image au travers de contours, (déterminant les frontières entre les objets) et de "textures". De manière très générale, un contour peut être défini par "le lieu de points connexes qui possèdent une forte transition d'intensité lumineuse ou de texture". Cette variation peut être caractérisée de différentes manières, en fonction de l'intensité de la variation.

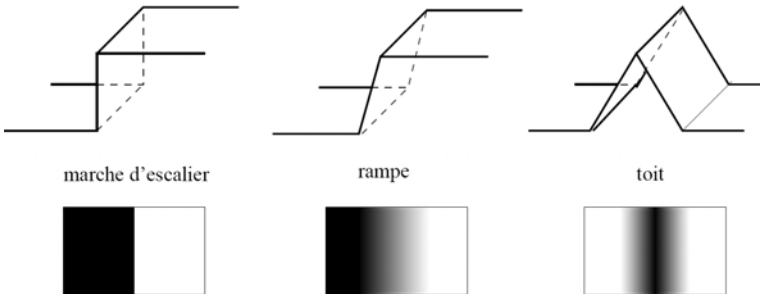

FIG. 2.6 – Différents type de contours

La figure 2.6 montre les différents types de contours sur base de l'observation de variation d'intensité. La figure appelée "marche" est caractéristique d'un changement brusque, on a un contour net. La seconde, ressemblant à une "rampe", apparaît lorsque la variation est plus progressive, le contour est flou. Enfin un contour caractérise par une courbe de variation en forme de "toit" apparaît lorsque cette variation est progressive mais décroît après un maximum, par exemple une ligne sur un fond uniforme.

Mes travaux ont tenté de détecter les contours de la cellule, et plus simplement du noyau. Plusieurs méthodes de contours ont donc été recensées pour leur soumettre nos images de globules blancs.

Les filtres de contour présentés ci-dessous sont basés sur une convolution. Celle-ci exprime le fait que l'on évalue la valeur d'un pixel à partir des pixels qui l'entourent.

Filtre de Sobel et de Prewitt : Les filtres de Prewitt et de Sobel sont des filtres basés sur une approche discrète d'une fonction : on parle de gradient de la fonction. En plus de cette dérivation, ces deux opérateurs de convolution introduisent un opérateur de "lissage", c'est-à-dire une opération visant à réduire les bruits qui faussent l'information de l'image. Cette opération, qui permet de "nettoyer" l'image pour les traitements ultérieurs, est réalisée au moyen de filtres linéaires passe-bas (cfr [Pee07]).

L'idée de ces filtres est basée sur les masques horizontaux et verticaux suivants :

$$G_x = \begin{vmatrix} -1 & 0 & 1 \\ -c & 0 & c \\ -1 & 0 & 1 \end{vmatrix} \qquad G_y = \begin{vmatrix} -1 & -c & -1 \\ 0 & 0 & 0 \\ 1 & c & 1 \end{vmatrix}$$

Prewitt et Sobel se différencient par la valeur de c :

– c = 1, pour un filtre de Prewitt ;
– c = 2, pour un filtre de Sobel.

Ces masques vont être "promenés" sur toute la matrice représentant l'image (excepté la première et dernière colonne ainsi que la première et dernière ligne). Le passage du masque va modifier la valeur du pixel en fonction de la variation d'intensité entre les pixels qui l'entourent. Ces deux masques permettent de faire apparaître les contours verticaux ou horizontaux en fonction de la matrice appliquée. Ces filtres sont moins sensibles au bruit. Le filtre de Sobel donne une meilleure estimation que celui de Prewitt car la série 1 2 1 est approximativement une gaussienne.

Filtre de Canny : L'algorithme de Canny est connu pour apporter de meilleurs résultats. Il contient plusieurs étapes dont une étape d'élimination du bruit et une étape appliquant un filtre de Sobel. Il n'est donc pas constitué que d'une convolution (c'est-à-dire l'application d'une matrice de transformation sur chacun des pixels de l'image). Cette particularité lui permet d'apporter de meilleures performances au niveau de la détection, la localisation et de l'unicité du contour.

Nous n'entrerons pas dans les détails de ces méthodes de détection de contours. Elles n'ont été utilisées que rapidement suite à des lectures portant sur la détection de contours et le temps a manqué pour approfondir la compréhension de leur fonctionnement. La figure 2.7 montre les résultats obtenus après l'application des différentes méthodes.

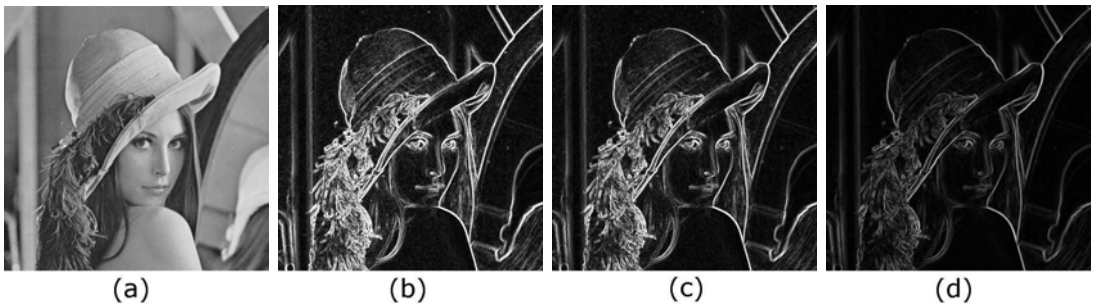

FIG. 2.7 – Image originale (a), image filtrée avec Sobel (b), image filtrée avec Prewitt (c) et image filtrée avec Canny (d)

2.5 Contenu du sang

2.5.1 Les globules rouges

Les globules rouges, comme introduits dans la partie 1.2, forment la plus grande partie du contenu du sang. Ils servent, entre autre, à transporter l'oxygène. Ils apparaissent, normalement, sous la forme d'un disque biconcave de couleur rouge (due à l'hémoglobine qu'ils contiennent). Ils ne disposent pas de noyaux et mesurent en moyenne 7 μm.

Ils peuvent se présenter sous d'autres formes, ce qui peut révéler des dysfonctionnements ou une pathologie particulière.

2.5.2 Les plaquettes

Les plaquettes ou thrombocytes sont des petits éléments du sang. Ils proviennent de la mœlle osseuse. Elles ne contiennent pas de noyaux. Elles sont de forme arrondie et de contour irrégulier. Leur diamètre va de 1 à 2μm, mais peut, pour de jeunes thrombocytes, atteindre 7μm. Leur couleur tend vers le gris clair dans lequel on observe des granulations de teinte rosée. Les plaquettes ne sont pas des cellules entières, elles apparaissent comme des fragments de cellules qui parfois se présentent sous forme d'agrégats.

2.5.3 Le plasma

Le plasma est le liquide dans lequel baigne l'ensemble des composants "solides" du sang. Il est constitué d'eau, de minéraux, de gaz, de nutriments (glucides, lipides,...), d'hormones et de déchets (urines, ...)

2.5.4 Les globules blancs

Les globules blancs sont les cellules issues de la mœlle osseuse et participent aux fonctions immunitaires du corps. Il existe plusieurs types de globules blancs, les principaux sont les neutrophiles, les éosinophiles, les basophiles, les lymphocytes et les monocytes.

Reprenons chacune de ces classes et décrivons-les dans les cas normaux :

Les neutrophiles : Leur taille varie de 12 à 14 μm. Leur noyau est plurilobé, de 3 à 5 lobes (c'est-à-dire qu'il est fragmenté). Ces lobes sont ronds ou ovales et reliés par des filaments qui ne sont pas toujours visibles. Dans certains cas, le noyau n'est composé que d'un lobe et d'un appendice (appelé corpuscule de Barr). La chromatine est dense et irrégulière, de couleur violet foncé.

Le cytoplasme, c'est-à-dire la partie de la cellule entourant le noyau, est de couleur beige rosé et contient de nombreuses granulations fines marron rosé.

Le ratio entre la taille de la cellule et celle du noyau est de 0,3. La figure 2.8 montre différents neutrophiles.

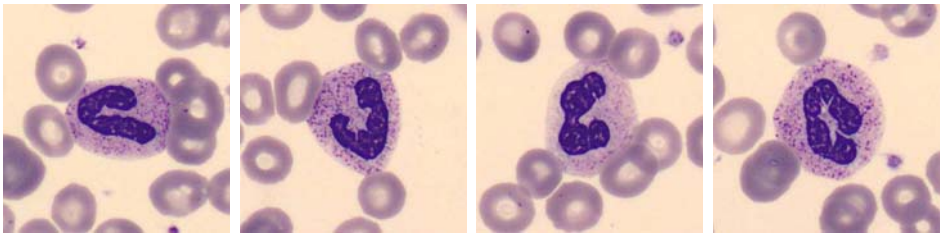

FIG. 2.8 – Neutrophiles

Les éosinophiles : Leur taille est égale à celle des neutrophiles (pour rappel, de 12 à 14 μm). Leur noyau est composé de deux lobes réunis par un "pont incurvé". La chromatine est dense et de couleur violette foncée.

Le cytoplasme est de couleur beige et contient des granulations sphériques (de 0,5 à 1,5 μm) réfringentes (c'est à dire qui réfractent la lumière). Ces granulations sont de couleur orangée et couvrent l'ensemble du cytoplasme comme dans "un sac de billes".

Le ratio entre la taille de la cellule et celle du noyau est de 0,3.

Les basophiles : La taille des basophiles va de 10 à 14 μm. Le noyau est volumineux et bilobé ou en forme de trèfles. Il peut aussi contenir des blocs de chromatine violet rouge foncé.

Le cytoplasme est rose clair et contient de nombreuses granulations spécifiques (de 0,2 à 1 μm) de couleur rouge pourpre ou violet noir. Ces granulations sont présentes partout dans la cellule et peuvent même recouvrir le noyau.

Le ratio entre la taille de la cellule et celle du noyau est de 0,5.

La figure 2.9 illustre différents basophiles.

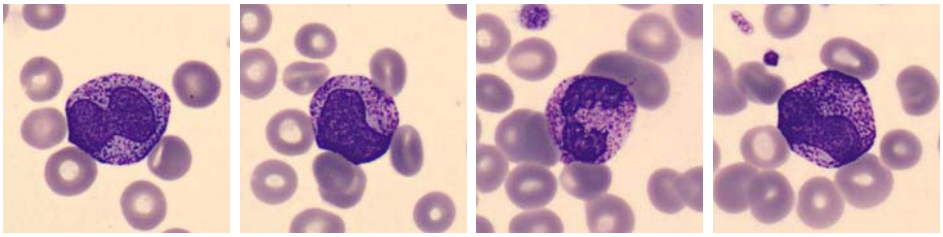

FIG. 2.9 – Basophiles

Les lymphocytes : Il existe deux types de lymphocytes.

- Les petits lymphocytes ont une taille variant entre 7 et 9 μm. Leur noyau est rond, ovale ou en forme de reins. La chromatine à l'intérieur de celui-ci est dense et de couleur violette foncée. On peut percevoir des masses chromatiniennes sombres. Le cytoplasme de la cellule est peu étendu, clair et de couleur légèrement bleutée.
- Les grands lymphocytes ont une taille allant de 9 à 15 μm. Leur noyau est positionné au centre de la cellule et de couleur violet foncé. Les masses chromatiniennes sont disposées en blocs et séparés par des zones plus claires. Le cytoplasme est plus étendu

que pour les petits lymphocytes. De couleur claire légèrement basophile, il peut contenir des petits grains intracytoplasmiques (3 à 5) de couleur rouge-violette.

Différents lymphocytes sont illustrés à la figure 2.10.

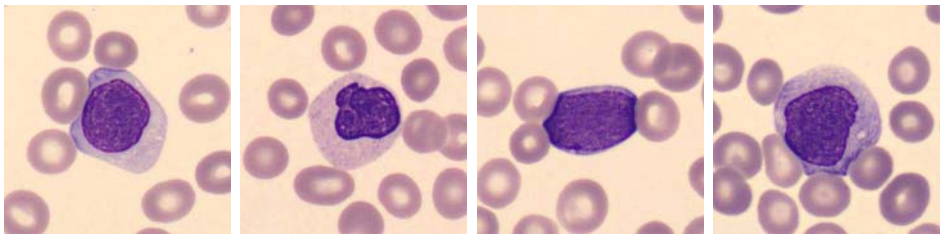

FIG. 2.10 – Lymphocytes

Les monocytes : Ils sont les plus grands, leur taille oscille entre 20 μm et 40 μm. La forme de leur noyau varie fortement, ils peuvent apparaître ronds, ovales, ou tout à fait irréguliers, avec des lobes ou découpés. La chromatine qu'il contient est peu dense, de couleur rouge violacée.

Le cytoplasme est, lui, très étendu et gris-bleu, il peut contenir des petites vacuoles et des petites granulations peu visibles. De manière générale, ils sont particulièrement difficiles à identifier, ils peuvent changer de forme et d'aspect.

Des monocytes sont illustrés à la figure 2.11.

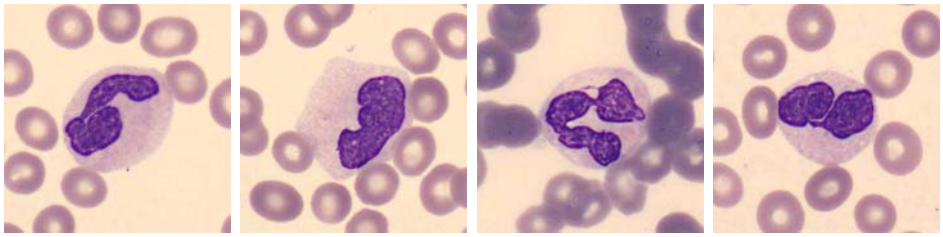

FIG. 2.11 – Monocytes

2.5.5 Arbre de classification

Un document fourni par les laborantins contient un arbre de classification reprenant la démarche de l'expert pour classifier un globule blanc. Cet arbre a une hauteur de maximum 5 niveaux.

Le premier niveau permet de faire une distinction entre les cellules contenant des grains ou pas. Si la cellule (le cytoplasme) contient de simples granulations, ou des grains plus gros, alors le second niveau propose 4 classes sur base de la basophilie du cytoplasme. La basophilie évoque le fait que l'image de la cellule laisse transparaître le colorant basique de la coloration des cellules. Pour la coloration des cellules sanguines, on utilise habituellement la coloration de May-Grünwald Giemsa, qui contient du bleu de méthylène.

- si la basophilie est élevée et que le noyau de la cellule est rond, alors on observe le nombre de grains : 2
  * si le nombre de grains est élevé, on a affaire à un *ProII* ;
  * s'il est peu élevé, c'est un *ProI*.
- si la basophilie est faible, que le noyau est rond ou ovale et que le rapport entre la cellule et son noyau est de plus ou moins 0,5, la cellule est un *myélocyte*
- si la basophilie est très faible, on observe le noyau :
  * si le noyau est allongé, c'est un *métamyélocyte* ;
  * s'il est rond, c'est un *lymphocyte à grains*.
- si la couleur de la cellule vire au gris, qu'il y a une présence de vacuoles ou de fins grains et que le noyau est variable, alors la cellule est caractérisée comme *monocyte*.

Dans le cas où la cellule ne renferme pas de grains, le deuxième niveau d'analyse est effectué sur base de la structure du noyau.

- si le noyau est plutôt fin et contient des vacuoles alors on classifie selon la teinte des vacuoles :
  * si elles sont pâles, la différenciation se fera sur base de leur nombre : 3 ou moins, c'est un *lymphoblaste*, plus de 3 un *myéloblaste*.
  * si les nucléoles sont plutôt bleutés, il s'agit d'un *proérythroblaste*.
- si le noyau est condensé, on observe le rapport entre la taille du noyau et celle de la cellule,
  * si le rapport est élevé, nous sommes en présence d'un *petit lymphocyte*.
  * dans les autres cas, ce rapport peut être fortement variable, cela peut être un *érythrocyte* (globule rouge), un *basophile*, un *pycnocyte* (forme de globules rouges), un *polycyte*, un *plasmocyte* ou encore un *lymphocyte activé*. L'arbre ne donne pas de précision sur les critères nécessaires pour différencier chacune de ces classifications.

Cet arbre (voir figure 2.12) apporte une première aide au classement des globules. Il fait déjà apparaître beaucoup de classes mais sans donner beaucoup de détails pour certaines d'entre elles. Il permet cependant de se rendre compte de la complexité du domaine et de centrer les développements sur certaines caractéristiques, telles que la forme de la cellule ou du noyau, ou encore la présence de grains.

Il serait intéressant aussi de pouvoir numériser les critères importants. En effet, certains d'eux sont très vagues. On parle de "rapport élevé", de "couleur virant au gris", de "basophilie élevée" ou encore de "rapport variable". Toutes ces données ne sont pas exploitables. Une observation statistique pourrait permettre de mettre en avant des valeurs caractéristiques.

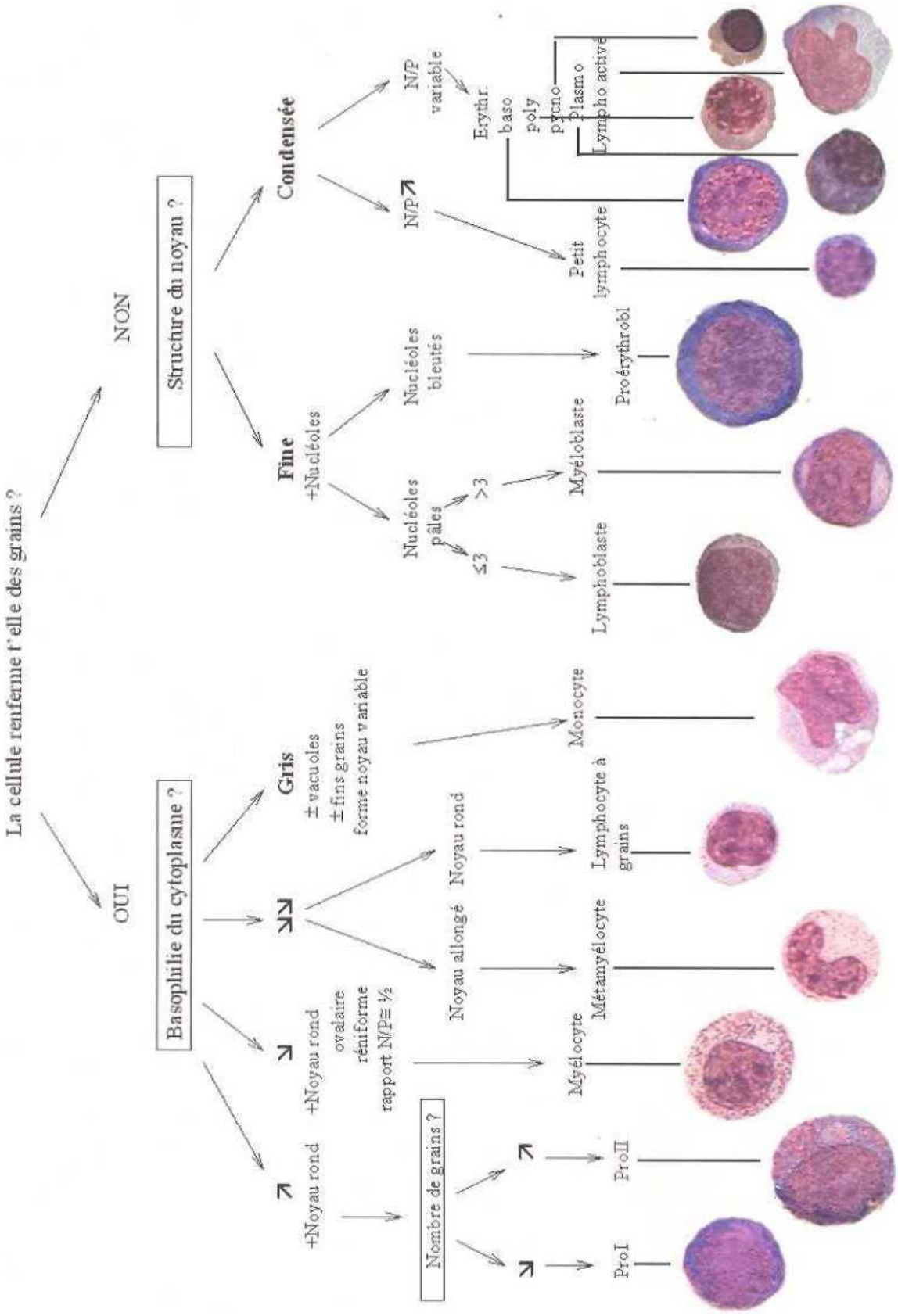

FIG. 2.12 – Arbre de classification de globules

2.6 Le standard "DICOM"

Avec l'avènement de la télé-médecine, il a fallu trouver un moyen d'organiser et de structurer la masse d'informations importantes issues de l'imagerie médicale. Il était nécessaire de disposer d'un standard permettant la gestion et la communication des informations archivées. C'est dans cette optique qu'est apparu le standard DICOM, pour *"Digital Imaging and COmmunications in Medicine"* (imagerie et communication numériques en médecine)[8].

Apparu en 1985, il avait pour but de standardiser les informations issues d'appareils d'analyses de constructeurs différents telles les images de CT (Computed Tomography), de MRIs (Magnetic Resonance Imaging), d'ultra-sons.

Ce format est bien plus qu'un simple format d'image. Il se différencie des images bitmap, JPEG, ou autres formats habituels par la possibilité de coupler à l'image des informations sur le patient ou l'identifiant de l'examen auquel l'image est liée.

Le format DICOM propose une série de champs permettant de lier des informations à une image. A chaque champ est lié un tag particulier. Pour illustrer ces propos, la figure 2.13 reprend certains des champs principaux utilisés.

Champ	Tag	Valeur
Scheduled Procedure Step Location	(0040,001)	Laboratoire d'hématologie
Pixel Data	(7FE0, 0010)	données de l'image
Institution Name	(0008,0080)	UCL Mont-Godinne
Station Name	(0008,1010)	Microscope Virtuel
Patient's Name	(0010,0010)	Nom du patient
Patient's Birth Date	(0010,0030)	Date de naissance du patient

FIG. 2.13 – Exemples de champs DICOM (issus de [Pee07])

Tous ces champs sont utilisés au travers du java et plus précisément lors de la création des objets java.

2.6.1 Utilisation au sein de l'établissement UCL - Mont-Godinne

Le standard DICOM est utilisé au sein des Cliniques Universitaires de Mont-Godinne dans le but de communiquer et de stocker les résultats des analyses. Les images issues de nombreuses analyses sont disponibles rapidement via le réseau interne. Il permet aux experts de disposer rapidement des résultats des examens et de pouvoir rapidement poser un diagnostic. L'utilisation du standard est couplée à un outil logiciel qui permet de visualiser les images d'un serveur, nommé "Telemis". Il propose une plateforme de visualisation d'images (distantes ou non) de tous les formats dont DICOM.

[8]from `Wikipediahttp://fr.wikipedia.org/wiki/DICOM`

2.7 La communication RS232

RS232 est un standard permettant la communication avec les périphériques connectés via un port série. Cette communication est de type asynchrone[9]. Cette caractéristique impose aux messages envoyés de contenir un tag informant le début d'une transmission (appelé bit START) et d'un autre informant la fin de la transmission (appelé bit STOP)([CCM]). La transmission se réalise de manière séquentielle, bit après bit[10].

FIG. 2.14 – Connecteur RS232 à 9 broches, prise mâle (à gauche), prise femelle (à droite)

La figure 2.14 ci-dessus montre les prises des ports caractéristiques du RS232.

Il existe plusieurs formes de port RS232, se différenciant par le nombre de broches du connecteur (9 ou 25 broches). Ce port a été progressivement remplacé par le port USB (suivi du USB2).

2.7.1 Utilisation

L'intérêt du standard RS232 apparaît lors de la mise en place de communication avec la platine du microscope et avec l'autofocus.

En effet, chacun de ces composants est relié à l'ordinateur de contrôle au moyen de connexions série.

Ce standard intervient, dans notre projet, dans un module java permettant d'envoyer les requêtes de commandes au microscope et à l'autofocus.

Un package java est disponible pour supporter la communication entre les différents éléments. Le package javax.comm appartient à Java Communications 3.0 API[11], il facilite le développement d'applications de communications telles que RS232 ou encore le port parallèle.

[9] Pour rappel, une communication asynchrone désigne une communication pour laquelle il n'y a pas de synchronisation. Les informations sont donc envoyées sans que les deux entités soient sur la même cadence.

[10] `http://fr.wikipedia.org/wiki/Port_s\%C3\%A9rie`

[11] `http://java.sun.com/products/javacomm/`

Chapitre 3

Analyses et Résultats

3.1 Introduction

Cette troisième partie va permettre d'exposer l'ensemble des résultats obtenus dans les quatre objectifs énoncés au point 1.6. Elle reprend donc les objectifs suivants :

- la mise en place d'une architecture supportant les erreurs de l'autofocus (point 3.2) ;
- la création de méga-images par coregistration d'images de sources différentes (point 3.3) ;
- le développement d'un système de classification modulaire captant les caractéristiques de globules blancs d'une méga-image et capable de les interpréter selon des règles définies en interne.
- la poursuite du développement d'une application d'extraction d'informations d'une base de données Microsoft Access provenant du logiciel *CellaVision*.

3.2 Autofocus

3.2.1 Introduction

Depuis des années, le microscope AX-70 du laboratoire d'hématologie de Mont-Godinne est équipé d'un autofocus. Seulement, comme certains de mes prédécesseurs l'ont fait remarquer, un problème assez important persistait et ralentissait l'étape de capture d'images sources.

En effet, l'autofocus était défaillant dans certaines situations particulières. Ainsi lorsque l'image sur laquelle on désirait faire le point ne comportait pas de variations de tons, l'autofocus ne trouvait pas son niveau de netteté (cfr figure 3.1) et provoquait une erreur. Il faut savoir que l'autofocus ne travaille pas sur l'ensemble de la zone observée. Il détermine une zone rectangulaire au centre de celle-ci (zone de mise au point). Elle sert de repère pour le calcul du focus idéal. L'autofocus capture des images en faisant varier la hauteur de l'objectif du microscope afin de trouver celle qui rend l'image la plus nette possible.

Il arrive que la zone de mise au point ne soit pas assez contrastée pour pouvoir différencier deux images capturées. Le microscope va donc effectuer sa recherche jusqu'à atteindre

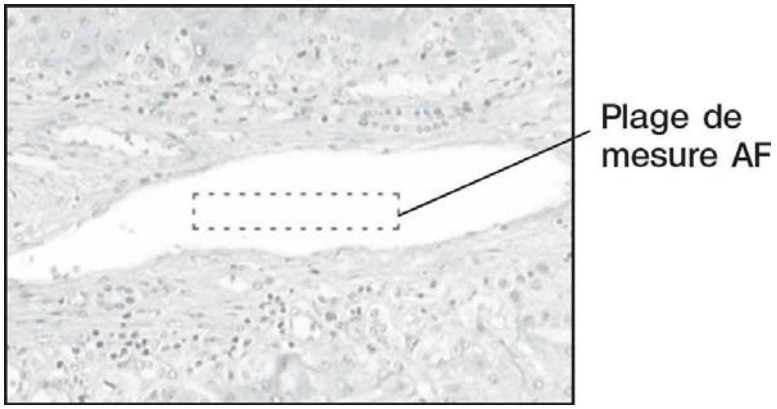

FIG. 3.1 – Exemple de situation où la plage de focus n'est pas assez contrastée.

les limites de l'axe Z de la platine. Une fois les limites atteintes, il fera savoir à l'utilisateur qu'il n'a pas réussi à trouver le niveau de netteté optimal par l'émission d'un message d'erreur. L'utilisateur doit alors intervenir manuellement pour mettre au point la zone à capturer. Il est évident que ce problème à un grand impact sur l'ensemble du processus de virtualisation automatique. Il oblige en effet l'utilisateur à s'assurer que l'autofocus est correctement réalisé et le cas échéant faire le point manuellement en ajustant la platine sur l'axe Z, pour pouvoir poursuivre la saisie. La capture devient semi-automatique puisqu'elle nécessite une présence continuelle.

Les travaux de mes prédécesseurs (notamment C. Peeters dans [Pee07]) ont envisagé deux pistes de solution. La première visait à utiliser un autofocus software intégré à Analysis. Ainsi lorsque l'autofocus hardware ne réussissait pas, le module Imaging-C faisait appel à la version software. Cependant, les résultats n'ont pas été satisfaisants, l'autofocus software effectuait des pas trop importants (de l'ordre de $0.2\mu m$ alors que $0.1\mu m$ aurait été plus significatif sur un objectif 100x).

La deuxième piste envisagée portait sur l'utilisation des fonctions U-IFRS propres à l'autofocus. Ces fonctions ("MOV", "POS" ou encore "AFC ?") permettaient d'interroger l'autofocus sur la position de celui-ci sur l'axe Z et d'ordonner l'autofocus à se placer à un niveau déterminé sur cet axe Z. Il aurait été possible de sauvegarder la position de la platine avant de lancer une mise au point et si celle-ci échouait, réajuster la platine à la position sauvegardée. La solution pouvait être encore améliorée en ajoutant une composante autofocus software (cfr [Pee07]).

A nouveau, cette seconde solution n'a pas fourni de résultats. En effet, les commandes fournies retournaient des messages d'erreurs. Il semble que le laboratoire ne dispose pas de la dernière version du firmware de l'autofocus. La version utilisée est trop ancienne et ne fournit pas encore ces commandes.

Au cours des discussions avec les représentants d'Olympus, d'autres pistes ont été proposées pour corriger ces erreurs. La première, qui consistait à effectuer une mise à jour du firmware, a tout de suite été abandonnée.

La seconde est semblable à la deuxième solution imaginée par C. Peeters à la différence qu'à la place de passer par l'autofocus, on utilise le microscope. Il a fallu l'équiper d'un moteur permettant de modifier la hauteur de la platine. Avant cela, un "inventaire" des appareils et de leurs connexions a aussi été réalisé pour permettre une bonne compréhension du système d'acquisition. Après avoir positionné le moteur, il a fallu mettre en place les composants logiciels permettant de faire interagir chacun des composants.

3.2.2 "Cartographie" des composants de l'installation du microscope.

Pour mettre en place la solution, il a fallu d'abord comprendre l'ensemble des connexions entre le microscope et l'autofocus ainsi que les différentes cartes d'acquisition et de communication de l'ordinateur de contrôle.

Le système mis en place pour la saisie automatique est composé de 9 appareils reliés entre eux :

L'ordinateur de contrôle : L'ordinateur de contrôle est l'hôte sur lequel Analysis Pro est installé et peut être lancé. Dans l'organisation du laboratoire, cet ordinateur accueille aussi le serveur de coregistration sur lequel sont stockées les images. L'ordinateur est relié au microscope via une cable parallèle.

Le microscope Olympus AX-70 : Le microscope (figure 3.2) est bien sûr le composant principal. Il est muni, entre autre, d'une platine permettant de recueillir la lame à observer et de différents objectifs (10x, 20x, 40x, 100x).

FIG. 3.2 – Le microscope Olympus AX-70

La caméra Sony PowerHAD DXC-950 (figure 3.3) : La caméra Sony est l'outil de capture d'images utilisé par Analysis. Elle est montée au-dessus de l'objectif du microscope AX-70. Il s'agit d'une caméra optique.

FIG. 3.3 – Caméra Sony PowerHAD montée sur le microscope.

Le clavier de contrôle du microscope : Ce clavier permet de modifier des propriétés du microscope telle que l'intensité lumineuse...

Le Sony Camera Adaptor CMA-D2 : Ce boîtier (n°1 sur la figure 3.4) situé au dessus de l'Olympus U-AFCB2 sert d'adaptateur pour la caméra Sony.

L'Olympus U-AFCB2 : Cet appareil (n°2 sur la figure 3.4) permet d'effectuer un autofocus sur le champ observé. Il est muni de multiples diodes dont les combinaisons permettent de se rendre compte de son bon fonctionnement. Il émet aussi des signaux sonores pour en attester.

L'adaptateur Olympus U-PS (Power Supply) : Cet outil (n°3 sur la figure 3.4) est relié au microscope AX-70 et au circuit électrique. Il fournit au microscope l'alimentation nécessaire à son fonctionnement.

L'interface Multi-Control 2000 : L'appareil Multi-Control 2000 (n°4 sur la figure 3.4) sert d'interface de conversion entre les appareils émetteurs et les composants moteurs montés sur le microscope. Il est connecté aux éléments générateurs de requêtes tels que l'ordinateur de contrôle et le joystick de contrôle, et aux éléments récepteurs effectuant les requêtes tels que les 3 moteurs agissant sur les 3 dimensions X,Y et Z (le moteur de l'axe Z est amovible).

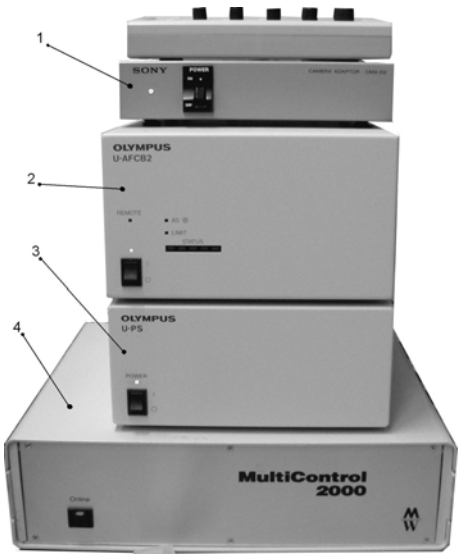

FIG. 3.4 – Bloc d'utilitaire comprenant l'U-AFC2

FIG. 3.5 – Pupitre Multi-Contrôle U-MCB

Le pupitre multi-commande Olympus U-MCB (figure 3.5) : Cet utilitaire multi-commande permet de contrôler les paramètres du microscope. Via un écran tactile, il offre la possibilité de modifier l'intensité de l'ampoule du microscope, l'objectif de visualisation,...

Le moteur amovible contrôlant l'axe Z de la platine (figure 3.6) : Ce composant est amovible. Il se monte sur l'axe Z du microscope et est relié au multi-control 2000. Ce moteur permet de déplacer la platine du microscope sur l'axe vertical.

FIG. 3.6 – Moteur contrôlant l'axe Z de la platine

FIG. 3.7 – Bloc de commande de mise au point

Le "bloc" de commande Olympus (figure 3.7) : Ce "bloc" de commande est aussi un appareil régulièrement utilisé. Il montre son utilité au moment de faire le point sur la zone observée. Le "bloc" est composé de 4 boutons sur le haut. Ceux-ci permettent entre autres de choisir une "politique" de focus ("one shot", "real time"). Mais la commande la plus importante est surtout la "molette" sur le côté gauche. Elle permet de modifier la hauteur (sur l'axe Z) de la platine et cela avec des "pas" plus petits que via le joystick ou la commande du microscope. On trouve aussi un bouton sur la face avant qui permet de modifier la taille du "pas" effectué.

Le joystick de contrôle (figure 3.8) : Le joystick est un outil très utilisé par les laborantins. Il permet de déplacer la platine de manière précise et rapide. Assez intuitivement, pour obtenir un mouvement de la platine vers la gauche ou la droite, on inclinera le joystick respectivement vers la gauche et la droite (même mécanisme pour le sens de la largeur). Pour déplacer la platine de haut en bas (sur l'axe Z), le joystick est muni d'une "tête" pivotante qui permet lorsqu'elle est soumise à un mouvement de gauche à droite de modifier la hauteur de la platine par rapport à l'objectif. Ce joystick est connecté directement au Multi-control 2000.

FIG. 3.8 – Joystick de contrôle de la platine du microscope AX-70.

La communication entre l'autofocus et l'outil de contrôle du microscope (plus précisément de la platine) s'effectue via une communication basée sur le protocole RS232.

L'ensemble de ces composants (figure 3.9) interagissent entre eux pour permettre la capture avec mise au point automatique. Certains d'entre eux proposent la même fonctionnalité, provoquant une certaine redondance.

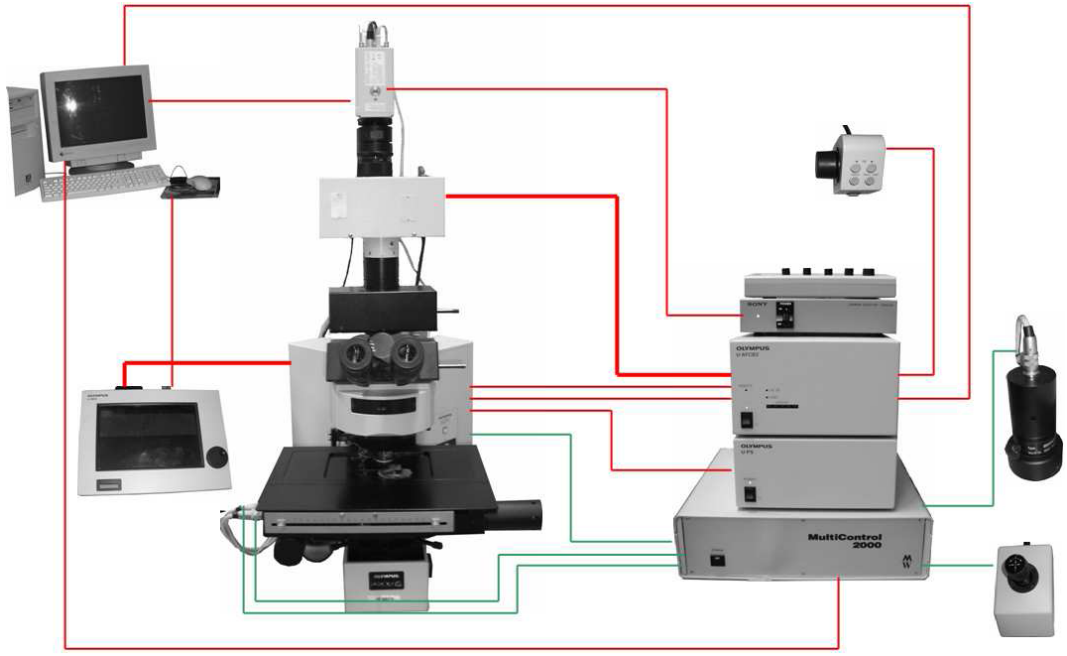

FIG. 3.9 – Cartographie des composants et de leurs connexions

3.2.3 Proposition de solution

Afin de solutionner ce problème récurrent, on doit mettre en place un mécanisme utilisant le protocole RS232. Celui-ci est en effet utilisé par l'ordinateur de contrôle pour communiquer avec le microscope et l'autofocus.

Étant donné que la version de l'autofocus ne dispose pas des fonctions permettant d'obtenir sa position ou de lui affecter une position, nous ne pouvions pas passer par cet appareil récalcitrant (cfr [Pee07]) et nous nous sommes adressés directement à la platine du microscope.

Pour cela, le laboratoire disposant d'un moteur permettant de piloter la platine pour la déplacer sur l'axe Z, nous l'avons installé sur le microscope.

Auparavant, il était possible de d'agir dans les trois dimensions par l'intermédiaire d'un boîtier de commande manuelle relié à l'autofocus (au moyen du bloc de commande à distance, cfr partie 3.2.2) ou via une "poignée pivotante" sur le côté du microscope. Cette dernière permet de déplacer la platine de haut en bas (ou inversement) selon "des grands pas" (on utilise alors la partie la plus à gauche de la poignée) ou "des petits pas" (on utilise la partie la plus à droite de la poignée).

Le microscope ne disposait pas encore d'outils supportant les commandes de déplacement la platine directement à distance (que ce soit d'un ordinateur de contrôle ou d'un outil de commande). Le joystick de contrôle n'autorisait que les mouvements sur les axes X et Y.

Le moteur doit se monter sur la poignée évoquée ci-dessus. Il est alors possible de piloter

la hauteur de la platine via deux canaux, soit en faisant pivoter vers la gauche ou la droite la "tête" du joystick pour influer sur le niveau de la platine, soit en utilisant les commandes fournies dans les manuels du microscope qui permettent à l'ordinateur d'obtenir la position de la platine mais surtout de modifier la hauteur (sur l'axe Z) de cette platine.

Un diagramme de séquence de la figure 3.10 permet de mieux visualiser la situation ainsi que la solution proposée.

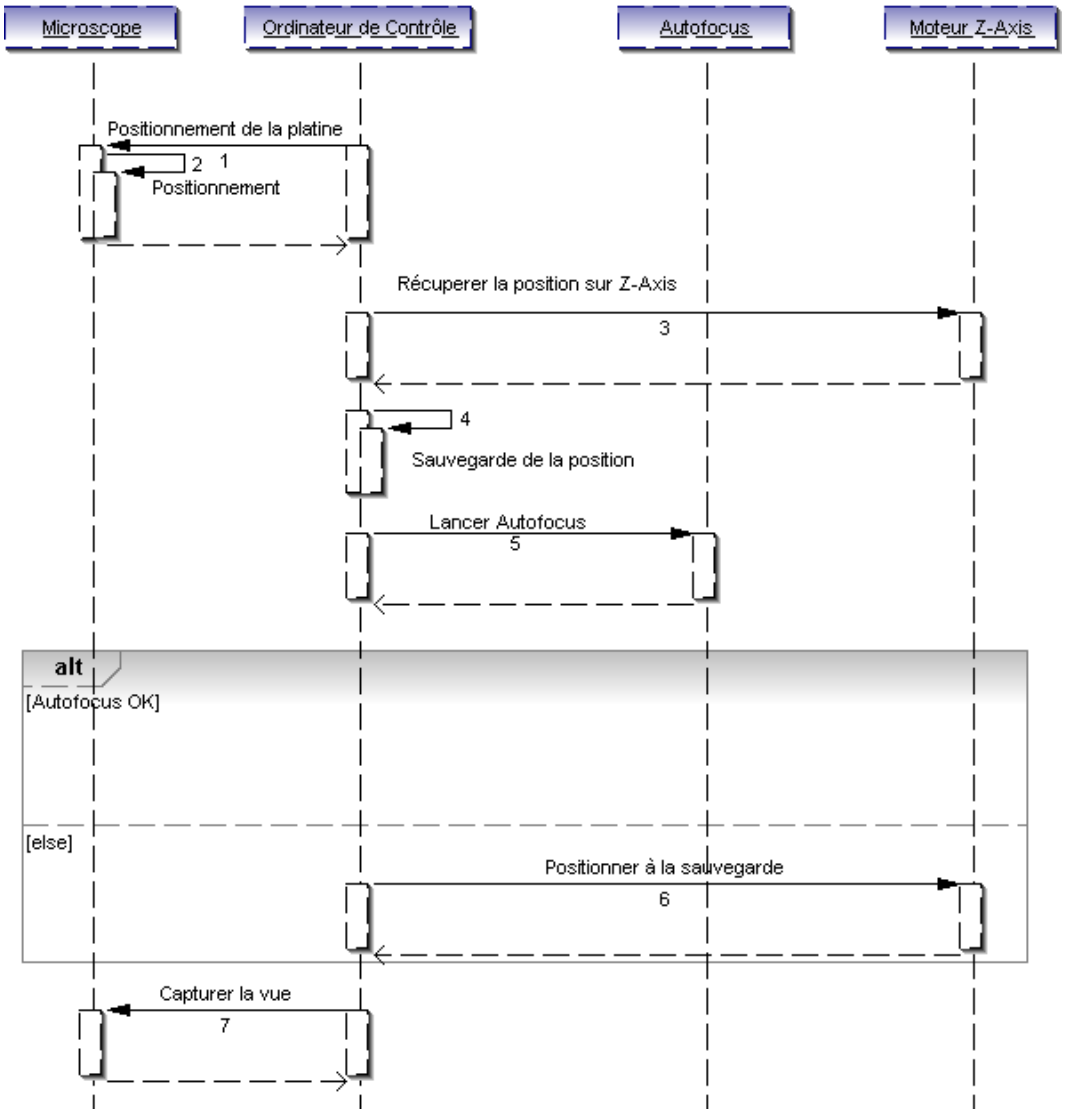

FIG. 3.10 – Diagramme de séquence illustrant la solution imaginée.

L'idée est d'encadrer les appels à l'autofocus par une sauvegarde de la position avant l'appel et la gestion des messages de retour.

L'algorithme mis en place se déroule en plusieurs étapes :

1. l'ordinateur envoie au microscope les coordonnées relatives (par rapport à la position au démarrage du microscope) de l'image à capturer ;
2. le microscope traite la requête et déplace la platine pour placer la lame en bonne position ;
3. l'ordinateur contacte le moteur de l'axe Z pour récupérer la valeur de la position sur l'axe Z ;
4. l'ordinateur sauvegarde la valeur récupérée dans un fichier de configuration ;
5. l'ordinateur active l'autofocus et récupère la valeur de retour ;
6. dans le cas où l'autofocus a répondu par un message d'erreur, il récupère la valeur précédemment sauvegardée et la communique au moteur de l'axe Z afin qu'il positionne la platine à son niveau initial ; si l'autofocus a réussi, nul besoin de repositionner la platine
7. on clôture le processus par la capture de l'image positionnée sous l'objectif.

3.2.4 Résultat et analyse de la solution

La seconde étape dans la mise en place de la solution a été de monter le moteur de l'axe Z sur le microscope (voir figure 3.11). Après avoir pris connaissance de chacune des pièces, nous avons pu les mettre en place grâce à quelques bricolages. Une fois en place, la poignée est entièrement recouverte et n'est donc plus utilisable. Les seules façons d'agir sur l'axe Z est d'utiliser le joystick de contrôle ou l'ordinateur de contrôle et son port série avec le MultiControl 2000.

FIG. 3.11 – Fixation du moteur sur le microscope.

Suite aux conseils de la société Olympus, nous nous sommes procuré différents utilitaires/manuels pour dialoguer avec la platine en utilisant les ports COM de l'ordinateur. Le logiciel "WinPOS" (voir la capture d'écran de la figure 3.12) disponible sur le site web `http://www.marzhauser.com/engl/i_soft.html` a donc permis de s'assurer que les commandes soient opérationnelles, qu'elles retournaient les bonnes informations. Les commandes telles que "m" ou "p", renseignées dans le manuel intitulé "Handbook Venus-1 for Corvus", se devaient de bien réagir pour aller plus loin dans la solution.

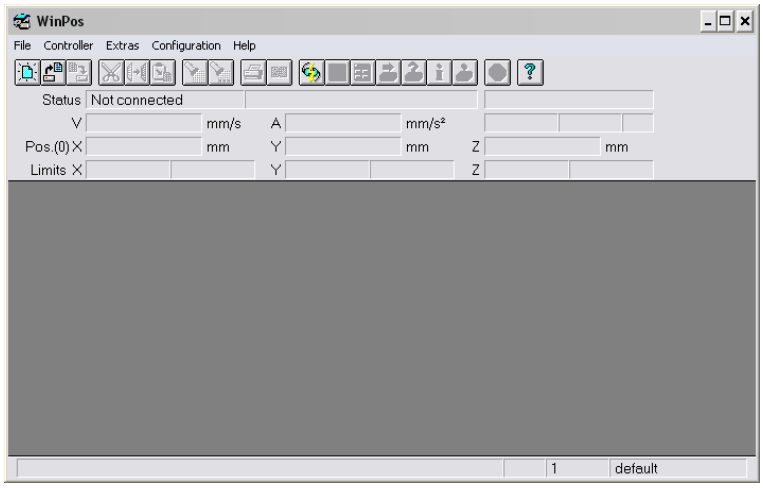

FIG. 3.12 – Capture d'écran de WinPos

Les tests de ces commandes ont donné de bons résultats. La platine se déplaçait correctement suite à l'utilisation de la commande "m" (ou "move") ou "r" (ou "rmove"). Ces dernières se différencient par le point utilisé pour le calcul de la nouvelle position : "r" permet un mouvement relatif, tandis que "m" situe à une position donnée par rapport au (0,0) originel. Ces commandes "m" s'utilisent avec 1, 2 ou 3 arguments en fonction du nombre de dimensions dont la platine a connaissance. Il faut donc préciser au microscope que l'on travaille sur 1, 2 ou 3 dimensions via la méthode "setdim". Pour notre objectif, nous travaillons en 3 dimensions (nous avons envoyé la commande "3 setdim" pour initialiser le domaine de travail).

En plus des commandes de positionnement, il fallait pouvoir récupérer la position courante. Pour cela, la commande "p"(ou pos) est fournie et retourne aussi une valeur correcte.

Les interactions entre l'ordinateur de contrôle et le moteur sont supportées par une communication RS232. Il a donc fallu implémenter une application Java permettant de communiquer directement avec la platine. Le package javax.comm propose des classes d'abstraction pour supporter les communications.

Le mécanisme est relativement simple à mettre en place et est fort proche de la gestion des sockets dans le cadre d'une architecture distribuée.

Un problème vite apparu est que l'autofocus ne parvient plus à fonctionner lorsque le moteur est en position. Il semble en effet qu'il y ait un conflit d'ordre physique entre l'autofocus et le moteur d'axe Z. Le moteur de l'axe Z monté sur la poignée pivotante du microscope empêche l'autofocus de fonctionner. Le premier conflit a été observé lors du démarrage du système. Le moteur ainsi monté sur la poignée a provoqué l'échec de l'initialisation de l'autofocus. Pour tenter de solutionner ce problème, le montage du moteur a été effectué après le démarrage de l'autofocus. Mais l'exécution de l'autofocus a rencontré un nouveau blocage. Le moteur, une fois monté, semble exercer une pression sur la molette et celle-ci empêche alors de modifier la hauteur en axe Z. L'autofocus, partageant le même

axe, n'a pas la capacité d'agir lorsque le moteur est monté sur le microscope.

Cette idée n'a pas apporté la solution attendue, elle n'a pas permis de contourner les erreurs générées par l'autofocus. Cependant, les recherches ont permis de comprendre le fonctionnement les uns par rapport aux autres des différents composants du microscope.

3.3 Coregistration d'images

Au cours de la phase de découverte et de prise en main des différents outils de travail, il est apparu que le laboratoire utilisait un autre type de caméra : une caméra *Olympus DP71* (cfr Figure 3.13).

FIG. 3.13 – Camera Olympus DP71

Cette caméra de 12 bits offre la possibilité de capturer en meilleure résolution. Elle était montée par les membres du laboratoire pour obtenir des images plus détaillées mais aussi plus grandes. De plus, cette caméra n'était pas branchée à la même station de travail que la caméra habituelle, *Sony PowerHAD*. Après avoir capturé les images, les laborantins n'utilisaient pas le serveur de coregistration développé par L. Zuyderhoff. Ils utilisaient un autre logiciel. Ajoutons que le temps de création d'une méga-image était bien plus important à cause de la capture effectuée manuellement. Dans la suite logique, un nouvel objectif est apparu : tester le serveur de coregistration construit par L. Zuyderhoff en fournissant des images sources d'une autre taille[1].

3.3.1 Premiers Résultats

Un premier problème est apparu avant même de lancer une coregistration. Il trouve son origine dans le fait que les "nouvelles" images sont (pour le moment) générées "à la main".

Il faut en effet savoir que le serveur de coregistration utilise en entrée des images sources dans le format de compression d'image ".tiff" (Tagged Image File Format) et dont

[1]Les images utilisées pour le test et les réglages étaient de 2040 pixels sur 1536 pixels, mais il semble que la caméra Olympus DP71 permette 4 résolutions d'images : (4080 x 3072), (2040 x 1536), (1360 x 1024), (680 x 512) cfr `http://olympusamerica.com/seg_section/product.asp?product=1016`

le format du nom de fichier est composé d'un préfixe commun et d'un suffixe lié à la position de l'image source par rapport à une matrice de coregistration.

Illustration : Pour illustrer le format des noms de fichiers, la matrice ci-dessous (figure 3.14) représente l'organisation des images sources dans la méga-image.

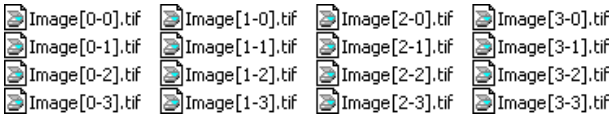

FIG. 3.14 – Format des fichiers générés par Analysis et pris en compte par le coregistreur

Les images capturées manuellement sont aussi enregistrées manuellement. L'utilisateur enregistre donc ces images sous un format d'image qui n'est pas nécessairement ".tif" et les images sources sont différenciées par un simple numéro à la fin du nom commun (un préfixe sur lequel il est possible de définir un ordre).

Illustration : La figure 3.15 ci-dessous , les images qui ont été fournies, portaient le nom de "ImageX.jpg" dont le nom est composé d'un préfixe commun à toutes les images sources (Image) et d'un suffixe étant un chiffre identifiant chacune d'elles (X). Ce chiffre est incrémenté d'une unité à chaque nouvelle image source créée.

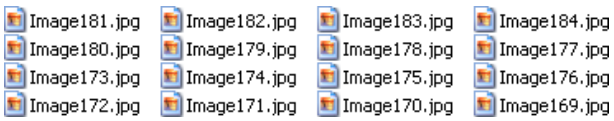

FIG. 3.15 – exemple de format de liste de fichiers à coregistrer

Ajoutons que ces images sont générées selon un schéma défini par l'utilisateur. Plusieurs schémas de génération sont envisageables et repris sur la figure 3.16 : ils sont trouvés par la combinaison du *haut vers le bas, bas vers le haut* et *gauche à droite, gauche à droite alterné, droite à gauche, droite à gauche alterné*. Cet ordre varie d'un utilisateur à l'autre, selon son ordre de capture des images sources.

En résumé, les fichiers générés manuellement ont donc deux grandes différences par rapport aux fichiers d'images sources générés automatiquement :

- la première différence porte sur le format du nom des fichiers des images sources ;
- la seconde différence porte sur le format de compression de l'image source.

V : haut en bas **H** : gauche à droite alterné	1 → 2 → 3 ↓ 6 ← 5 ← 4 ↓ 7 → 8 → 9	**V** : haut en bas **H** : gauche à droite	1 → 2 → 3 ↙ 4 → 5 → 6 ↙ 7 → 8 → 9
V : haut en bas **H** : droite à gauche alterné	3 ← 2 ← 1 ↓ 4 → 5 → 6 ↓ 9 ← 8 ← 7	**V** : haut en bas **H** : droite à gauche	3 ← 2 ← 1 ↘ 6 ← 5 ← 4 ↘ 9 ← 8 ← 7
V : bas en haut **H** : droite à gauche alterné	9 ← 8 ← 7 ↑ 4 → 5 → 6 ↑ 3 ← 2 ← 1	**V** : bas en haut **H** : gauche à droite	9 ← 8 ← 7 ↗ 6 ← 5 ← 4 ↗ 3 ← 2 ← 1
V : bas en haut **H** : gauche à droite alterné	7 → 8 → 9 ↑ 6 ← 5 ← 4 ↑ 1 → 2 → 3	**V** : bas en haut **H** : gauche à droite	7 → 8 → 9 ↖ 4 → 5 → 6 ↖ 1 → 2 → 3

FIG. 3.16 – Différentes combinaisons pour l'ordre de génération des images sources

Pour remédier à ces différences, une application a donc été développée pour convertir le format (de nom et de fichier) des images sources pour qu'elles soient acceptées par le serveur de coregistration. Cette application demande des arguments :

- l'ordre horizontal de génération (gauche à droite alterné, gauche à droite, droite à gauche alterné, droite à gauche) ;
- l'ordre vertical de génération (haut vers bas ou bas vers haut) ;
- le préfixe commun à toutes les images sources ;
- le format des images à créer ;
- le chemin d'accès à la première image à convertir ;
- le nombre d'images sources en hauteur dans la méga-image ;
- la largeur en nombre d'images sources de la méga-images ;

Les valeurs des constantes pour les arguments de l'application de conversion sont :

- pour le sens horizontal de génération,

  - de droite à gauche, 0 ;
  - de gauche à droite, 1 ;
  - de droite à gauche alterné, 2 ;
  - de gauche à droite, 3 ;
- pour le sens vertical de génération,
  - de haut en bas, 4 ;
  - de bas en haut, 5 ;

Pour illustrer la configuration des paramètres, reprenons les deux illustrations précédentes. Ainsi, en partant des fichiers illustrés dans la figure 3.15, on veut obtenir les fichiers de la figure 3.14. Les arguments auront donc les valeurs :

* l'argument pour l'ordre horizontal de génération sera égal à 2 (constante pour exprimer que les images sont générées de droite à gauche en alternance) ;
* l'argument pour l'ordre vertical de génération sera égal à 5 (constante pour exprimer que les images sont générées du bas vers le haut) ;
* le préfixe commun à toutes les images sources vaudra "Image" ;
* le format d'image à créer est "tif" ;
* le chemin d'accès à la première image à convertir (Sous Windows, par exemple, "C :/Image/Image169.jpg" ;
* le nombre d'images sources en hauteur dans la méga-image sera égal à 4 ;
* la largeur en nombre d'images sources de la méga-images sera égal à 4 ;

Une fois, les paramètres bien configurés, il suffit que l'utilisateur exécute cette application de conversion avant de lancer le serveur de coregistration et le client relatif. Elle se charge de convertir les noms des images sources, mais aussi le format de compression. En effet, elle contient les codecs pour transformer les images .jpeg (ou jpg ou encore .JPEG)[2], .png (ou PNG)[3], .pnm (ou PNM)[4] et .tif (ou .tiff ou encore .TIF)[5].

Le premier test a été réalisé sur le serveur utilisé pour les images créées avec la caméra Sony PowerHAD, aucun des paramètres n'a été modifié. Les résultats obtenus n'ont pas été très encourageants (cfr annexe 6.2). En effet, les images ne sont pas du tout bien positionnées et sont assez loin de leur position optimale.

La deuxième étape a consisté en la modification des paramètres de coregistration se trouvant dans le fichier de configuration du serveur.

Cette étape ne donnant pas de meilleurs résultats, l'analyse du code s'est imposée. Lors de la relecture, il est apparu que beaucoup d'autres paramètres entraient en compte dans la coregistration. Ces paramètres étaient propres aux images de 768 pixels sur 573 pixels. Il a été décidé de sortir tous les paramètres utilisés qui, jusqu'alors, étaient "hard codés" et de les ajouter au fichier de configuration du serveur.

[2]Acronyme de **J**oint **P**hotographic **E**xperts **G**roup
[3]Acronyme de **P**ortable **N**etwork **G**raphics
[4]Acronyme de Portable anymap
[5]Acronyme de **T**agged **I**mage **F**ile **F**ormat

Plusieurs tests ont été réalisés mais aucun ne fut fructueux. Les paramètres n'étaient pas indépendants de la résolution des images sources. Lorsqu'un de ceux-ci fournissait une bonne coregistration horizontale, il n'était pas valable pour la coregistration verticale. Et vice versa.

Il a alors fallu comprendre d'où venait l'erreur, via une analyse du code et des résultats.

Muni du mémoire de L. Zuyderhoff ([Zuy03]) comme documentation, nous avons pu mettre en évidence certaines erreurs.

La première erreur mise en évidence, est une erreur sans influence sur la réussite d'une coregistration d'images de sources différentes. Elle peut diminuer la précision de l'application. Elle apparaît lors de la troisième étape de coregistration (voir point 2.3.3), plus particulièrement lors de la recherche du vecteur propre dans l'image esclave.

Rappelons les étapes :

Une fois la zone de recherche dans l'image esclave déterminée, au départ de chacun des pixels de cette zone, on va prélever la colonne (ou la ligne) d'une taille de buffer (de taille 493) (voir section 2.3.3) pour les comparer au vecteur étalon. Pour rappel, la zone est délimitée par le rectangle formé par les points (34,5), (94,5), (34,85), (94,85)

Dans le cas d'une coregistration horizontale, pour tous les pixels situés entre les lignes 5 et 80, aucun problème ne se pose. Mais pour les lignes entre 80 et 85, le problème apparaît car la démarche consiste à extraire de l'image esclave un tableau de pixel d'une taille de 493 pixels. Or, en observant la position du dernier pixel du tableau extrait, celui-ci reprend des pixels qui ne sont pas dans l'image. En effet, lorsque l'on calcule la position du dernier pixel du tableau pour la coordonnée (94,85) par exemple, elle se situe à une coordonnée en y égale à $5 + 80+$ la taille du buffer c'est-à-dire $5 + 80 + 493 = 578$ autrement dit, 5 pixels en dehors de l'image (d'une hauteur de 573). Dans ce cas, les 5 derniers pixels seront initialisés à 0, ce qui peut provoquer le rejet des colonnes concernées. Ainsi les colonnes, dont les pixels se trouvent sur les lignes de 81 à 85, ne seront jamais sélectionnées comme correspondantes au vecteur étalon.

Une deuxième erreur a été détectée et influence la réussite de la coregistration. Afin de bien comprendre les situations dans lesquelles apparaît cette erreur, rappelons les étapes du processus de coregistration de la section 2.3 :

- Lors de la première étape, un vecteur étalon issu de l'image maître est prélevé ;
- La deuxième étape consistait à déterminer la zone de recherche dans l'image esclave ;
- La troisième étape est celle qui va trouver la position du vecteur étalon dans l'image esclave ;
- La quatrième étape se charge de calculer la position du vecteur étalon dans l'image esclave.

Une analyse couplée à l'observation des différentes valeurs calculées a permis de mettre en avant les limitations de ce système. En effet, le système montre ses limites lorsqu'il désigne un vecteur étalon dans l'image "maître" (dans l'étape décrite au point 2.3.1, dite de sélection du vecteur étalon) qui ne se retrouve pas dans l'image "esclave" (lors de l'étape

décrite au point 2.3.3 dite de coregistration). Cette erreur est liée à la définition des bornes de la zone de recherche (voir la partie 2.3.2). Cette zone est déterminée arbitrairement et lorsque la superposition entre deux images est plus importante, il peut arriver que le vecteur étalon ne se situe pas dans la zone de recherche déterminée. L'illustration suivante permet de mieux visualiser le problème.

Illustration : Pour illustrer cette erreur, la figure 3.17 montre la colonne choisie dans l'image source maître (à gauche) et sa position dans l'image esclave (à droite). On peut clairement voir que le vecteur étalon prélevé dans l'image maître ne trouvera pas de correspondance dans l'ensemble des vecteurs de pixels verticaux extraits à partir des pixels de la zone de recherche (représentée par le rectangle rouge dans le grossissement de l'image esclave). Pour rappel, le vecteur étalon est le vecteur de pixels vertical prélevé dans l'image maître à partir du pixel choisi (de couleur noire dans le zoom de l'image maître ci-dessous). Autrement dit en restreignant le domaine à un seul pixel, le pixel noir de l'image maître ne se trouve pas dans la zone de recherche. Vous trouverez en annexe 6.1.1 le résultat de cette coregistration.

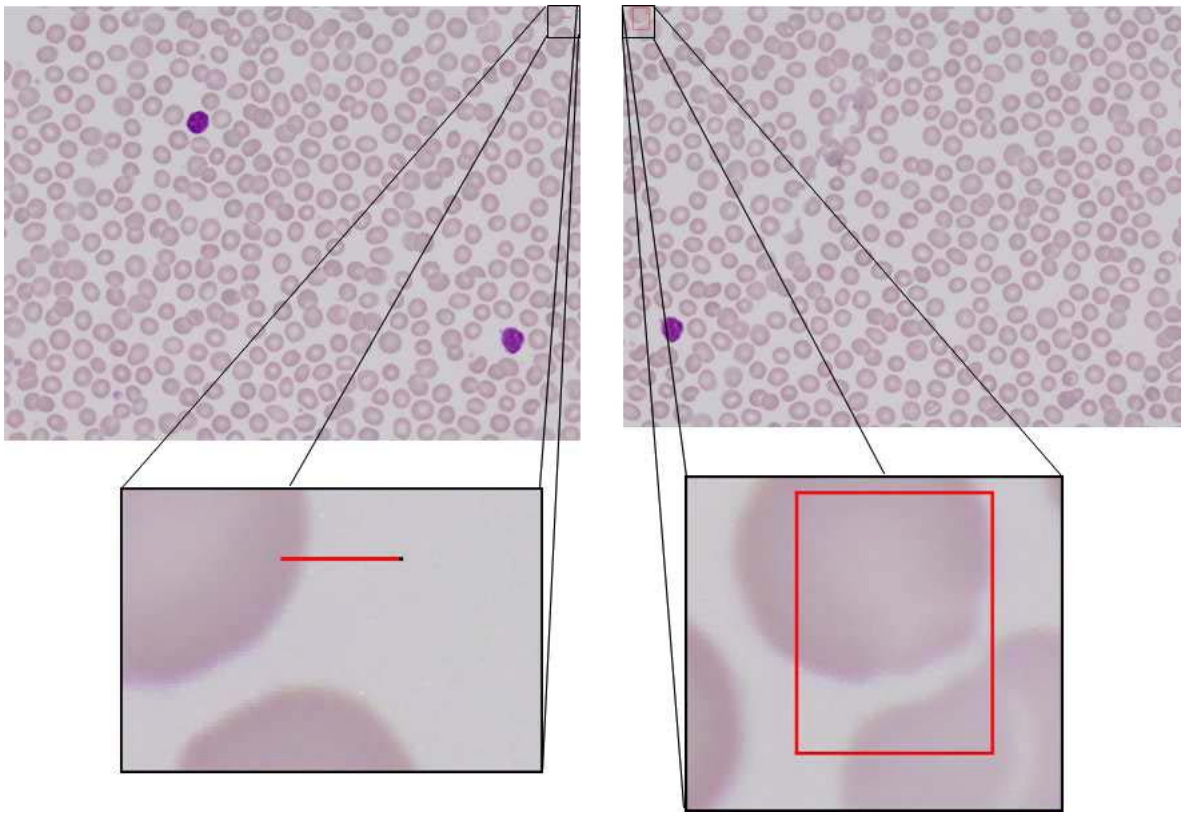

FIG. 3.17 – Illustration du choix d'un point hors zone de recherche.

3.3.2 Modification de la méthode de coregistration

Une fois la source d'erreur identifiée, deux pistes pouvaient être envisagées. L'une était de modifier les valeurs hard-codées, l'autre de poser une réflexion sur une nouvelle approche de coregistration. La première idée n'a pas été privilégiée. Des valeurs hard-codées ou passées en fichier de configuration ne fournissent pas une solution très souple.

La réflexion a donc été portée sur la refonte des mécanismes de coregistration. La solution présentée ne repose pas sur une révolution du cœur du serveur de coregistration. Les principes sont toujours d'application. Les changements apparaissent au niveau du choix des valeurs.

Plusieurs problèmes sont apparus. Ils sont principalement issus de la construction trop dépendante de la résolution des images et des caractéristiques de leur capture.

En effet, l'algorithme contient beaucoup de constantes hard-codées pour la bonne réalisation de la coregistration. Mais ces valeurs ne correspondent pas aux besoins de la coregistration d'autres types d'images. Pour illustrer les propos tenus, reprenons l'algorithme décrit au point 2.3. Au point 2.3.1, l'algorithme fait intervenir une constante 100 utilisée pour définir la valeur de recouvrement (overlap) des deux images ; une autre, 5, utilisée pour éviter que la coregistration ne prenne en compte des bords blancs qui pourraient provoquer des erreurs dans les calculs des valeurs ; une autre, 40.

Un autre problème cité plus haut, est lié au choix du vecteur étalon et de la zone de recherche. La zone de recherche déterminée n'est pas suffisamment grande, elle ne contient pas le vecteur étalon.

Les images utilisées jusqu'à présent, subissaient un décalage vers le bas à chaque pas vers la droite. Toutes les images ne subiront pas ce même décalage, encore moins pour les images générées à la main.

Pour solutionner cette erreur, la méthode a été repensée. La démarche reste la même mais le choix des "bornes" pour les différents intervenants du calcul (vecteur étalon, zone de recherche et décalage relatif) a été modifié.

Ainsi la méthode énoncée dans la partie 2.3 a été remaniée principalement dans l'étape de choix de zones de traitement, le nouvel algorithme comporte les mêmes étapes que le précédent :

1. **l'étape appelée *"de sélection du vecteur étalon"* (dans l'image maître) :** cette étape détermine le vecteur étalon dans l'image maître, c'est-à-dire la chaîne de pixels comportant le plus d'intérêt.
2. **l'étape appelée *"de sélection de la zone de recherche"* (dans l'image esclave) :** cette seconde étape consiste en le calcul de la position de la zone de recherche du tableau de pixels correspondant au vecteur étalon.
3. **l'étape *"de comparaison"* :** cette troisième étape calcule un indice de comparaison entre le vecteur étalon (issu de l'image maître) et les vecteurs à comparer (issus de l'image esclave). Il en résulte la position dans l'image esclave du vecteur à comparer le plus semblable au vecteur étalon .
4. **l'étape appelée *"de positionnement relatif"* (par rapport à l'image maître) :** cette étape clôt le processus de coregistration en calculant le décalage de l'image

esclave par rapport à l'image maître sur base des décalages issus de l'opération précédente.

Quelques paramètres ont été ajoutés au fichier de configuration du serveur. Ces paramètres ont pour but d'ajuster les zones d'intérêts. Reprenons l'ensemble des paramètres intervenant dans la coregistration :

- LAG_MAX_HORI permet de définir le décalage **horizontal** qu'il peut exister entre deux images à coregistrer **verticalement** ;
- LAG_MAX_VERT permet, de la même manière, de définir le décalage **vertical** qu'il peut exister entre deux images à coregistrer **horizontalement** ;
- OVERLAP_MAX_HORI définit un décalage maximum sur deux images à coregistrer **horizontalement** ;
- OVERLAP_MAX_VERT définit un décalage maximum sur deux images à coregistrer **verticalement** ;
- MAX_SEARCH_SPACE permet de définir un espace maximum à traiter pour la recherche du vecteur étalon ;
- MIN_SEARCH_SPACE permet de définir un espace minimum à ne pas traiter sur les bords des images (exemple : si les images comportent des bords blancs de 5 pixels comme sur la figure 3.18) ;
- FACTOR_SEEK_ORIGINAL permet de définir un facteur qui sera le diviseur de la hauteur (ou de la largeur si on coregistre horizontalement) lorsque le décalage maximum ne peut pas être déterminé. Il est utilisé pour limiter le processus de recherche de correspondance si les décalages sont non déterminés (valeur -1).

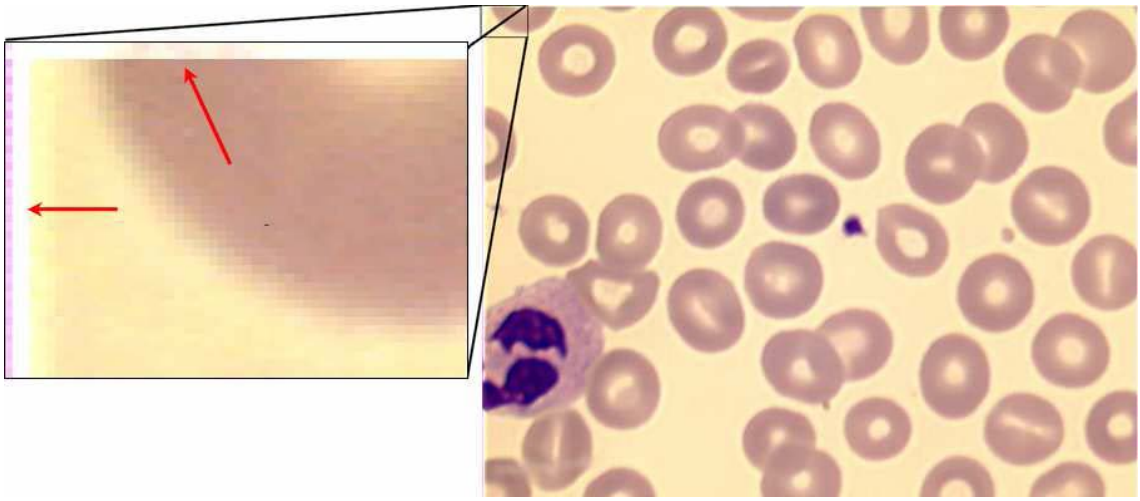

FIG. 3.18 – Exemple de bords blancs pouvant apparaître sur les images à coregistrer

Pour les développements suivants, nous considérerons les notations suivantes :

Soit

M_{ij},	la valeur du pixel de l'image M aux coordonnées (i,j)
M_i	la colonne à l'index i
l	la largeur de l'image
h	la hauteur de l'image
$BufV$	un tableau vertical de pixels noirs
v_{buff}	la longueur de $BufV = (l - (2 * \text{LAG\_MAX\_VERT}))$
$BufH$	un tableau horizontal de pixels noirs
h_{buff}	la longueur de $BufH = (h - (2 * \text{LAG\_MAX\_HORI}))$
LAG_MAX_HORI	le décalage horizontal entre deux images côte à côte verticalement
LAG_MAX_VERT	le décalage vertical entre deux images côte à côte horizontalement
MAX_SEARCH_SPACE	l'espace maximum de recherche
MIN_SEARCH_SPACE	l'espace minimum de recherche
OVERLAP_MAX_HORI	le recouvrement entre deux images côte à côte horizontalement
OVERLAP_MAX_VERT	le recouvrement entre deux images côte à côte verticalement
FACTOR_SEEK_ORIGINAL	facteur pour le calcul du recouvrement par défaut

Modification à l'étape de sélection du vecteur étalon

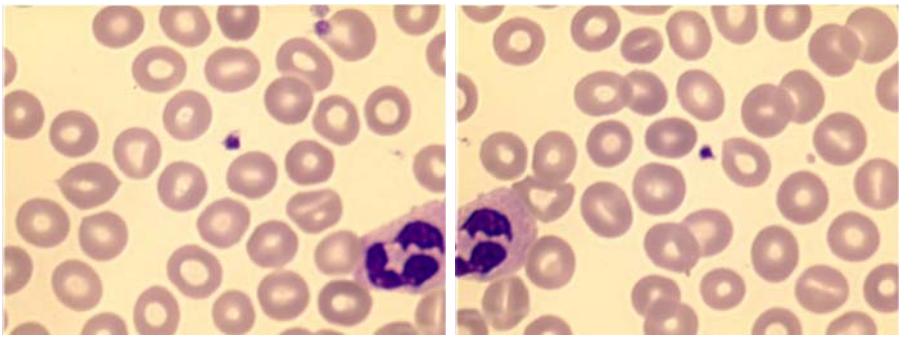

FIG. 3.19 – Deux images à coregistrer, l'image maître (à gauche) et l'image esclave (à droite)

La principale modification lors de la première étape est la modification de la taille du vecteur étalon. Comme dans la méthode présentée au point 2.3.1, la colonne (ou ligne) de pixels la plus "contrastée" est extraite de l'image maître (en se basant sur un vecteur de pixels noirs). Mais la taille de la zone dans laquelle on va prélever ce vecteur étalon est plus large. Cette modification permet de disposer de plus de choix et d'obtenir un "meilleur" vecteur étalon.

Ce choix se déroule en plusieurs étapes :
De nouveau, pour la coregistration horizontale :

1. détection d'une zone dans laquelle sera choisi ce vecteur étalon. Cette zone de sélection est délimitée par un "rectangle" horizontal (on parle de vecteur plus précisément) :
   - de hauteur $h_{ms} = 1$ pixel ;
   - de largeur $l_{ms} = (\text{MAX\_SEARCH\_SPACE} - \text{MIN\_SEARCH\_SPACE})$ pixels ;
   - positionné à une distance $y_{ms} = \text{LAG\_MAX\_HORI}$ pixels du bord supérieur ;
   - et à une distance $x_{ms} = (l - \text{MAX\_SEARCH\_SPACE})$ du bord droit.
2. Nous allons rechercher le vecteur t_i le plus contrasté parmi les l_{ms} colonnes de taille v_{buff} de cette zone. Nous avons déjà mis en évidence le problème venant du calcul de la norme du vecteur BufV. Pour réellement obtenir le vecteur le plus contrasté, nous pouvons utiliser la moyenne des niveaux de gris de t_i :

   $$E(t_i) = \frac{1}{v_{buff}} \sum_{j=0}^{v_{buff}-1} t_{ij}$$

   et la variance :

   $$var(t_i) = \frac{1}{v_{buff}} \sum_{j=0}^{v_{buff}-1} (t_{ij} - E(t_i))^2$$

3. Parmi toutes les variances, on choisit celle de valeur la plus grande, soit i^* et on note ve, le tableau de pixels de référence, **vecteur étalon**, dans l'image maître ($ve = t_i^*$). La coordonnée de son premier pixel est (x_{ve}, y_{ve})[6]

Et pour la coregistration verticale :

1. détection d'une zone dans laquelle sera choisi ce vecteur étalon. Cette zone de sélection est délimitée par un "rectangle" vertical
   - de hauteur $h_{ms} = \text{MAX\_SEARCH\_SPACE} - \text{MIN\_SEARCH\_SPACE})$ pixels ;
   - de largeur $l_{ms} = 1$ pixel ;
   - positionné à une distance $x_{ms} = \text{LAG\_MAX\_VERT}$ pixels du bord droit ;
   - et à une distance du bord supérieur $y_{ms} = (h - \text{MAX\_SEARCH\_SPACE})$.
2. Comme pour le cas horizontal, nous allons rechercher le vecteur t_i le plus contrasté parmi les h_{ms} colonnes de taille h_{buff} de cette zone. Le problème de la norme de BufV égale à 0 apparaît aussi pour BufH. Nous utilisons aussi la moyenne des niveaux de gris de t_i :

   $$E(t_i) = \frac{1}{h_{buff}} \sum_{j=0}^{h_{buff}-1} t_{ij}$$

   et la variance :

   $$var(t_i) = \frac{1}{h_{buff}} \sum_{j=0}^{h_{buff}-1} (t_{ij} - E(t_i))^2$$

[6]Donc, dans le cas horizontal, $y_{ve} = y_{ms}$, et $x_{ms} \leq x_{ve} < x_{ms} + l_{ms}$. Dans le cas vertical, $x_{ve} = x_{ms}$, et $y_{ms} \leq y_{ve} < y_{ms} + h_{ms}$

3. De la même manière que dans le cas horizontal, parmi toutes les variances, on choisit celle dont la valeur est maximale, soit i^* et on note ve, le tableau de pixels de référence, **vecteur étalon**, dans l'image maître ($ve = t_i^*$). La coordonnée de son premier pixel est toujours (x_{ve}, y_{ve}).[7]

Concrètement, les changements apparaissent dans le choix des emplacements et dimensions de chacun des tableaux (ou vecteurs) qui interviennent dans la méthode. Ce qui apporte une certaine souplesse et rend le système de coregistration indifférent à la résolution des images sources. Le serveur peut aussi, à présent, coregistrer des images sur un axe horizontal dont le décalage vertical est négatif ou positif. De même, pour une coregistration verticale, le décalage horizontal peut être positif et négatif. Nous verrons dans la suite que ces changements ont aussi un impact sur les étapes suivantes.

Illustration : Pour illustrer le développement ci-dessus, la figure 3.20 reprend l'image maître de la figure 3.19 et met en évidence la zone désignée. Précisons que les images ont une taille de 768 sur 573 et que les paramètres utilisés ont les valeurs :

- LAG_MAX_HORI =100
- LAG_MAX_VERT=100
- OVERLAP_MAX_HORI=-1
- OVERLAP_MAX_VERT=-1
- MAX_SEARCH_SPACE=50
- MIN_SEARCH_SPACE=6
- FACTOR_SEEK_ORIGINAL=3

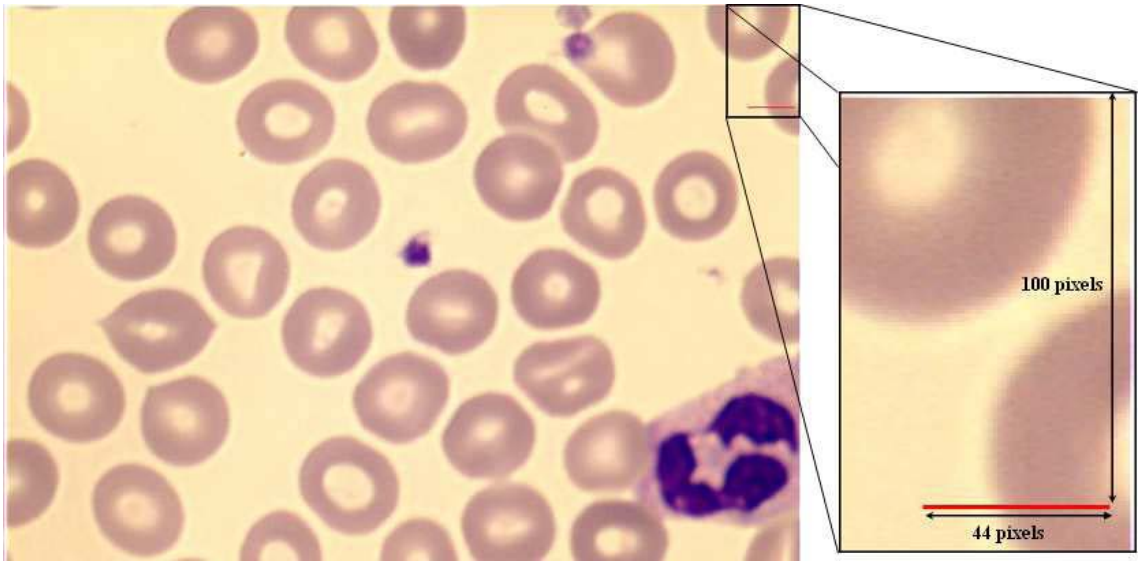

FIG. 3.20 – Détection du tableau de pixel témoin dans l'image maître

[7]Donc, dans le cas horizontal, $y_{ve} = y_{ms}$, et $x_{ms} \leq x_{ve} < x_{ms} + l_{ms}$. Dans le cas vertical, $x_{ve} = x_{ms}$, et $y_{ms} \leq y_{ve} < y_{ms} + h_{ms}$

Pour chacun des pixels rouges, on extrait une colonne de taille 373 que l'on compare au tableau de 373 pixels noirs. La colonne choisie est celle dont les coordonnées du premier pixel est $(761, 100)$

Modification de l'étape de sélection de la zone de recherche

La seconde opération consiste en la sélection de la zone de recherche dans l'image esclave (cfr section 2.3.2). Cette zone déterminera l'ensemble des pixels à partir desquelles seront prélevées les colonnes (lignes) si la coregistration est horizontale (verticale).

La zone de sélection est délimitée par l'algorithme de la manière suivante :

Dans le cas horizontal,

- La position du coin supérieur gauche, notée (x_{slave}, y_{slave}), est donnée par :
  - $x_{slave} =$ MIN_SEARCH_SPACE ;
  - $y_{slave} =$ MIN_SEARCH_SPACE ;
- La largeur de cette zone, si OVERLAP_MAX_HORI$\neq -1$, alors OVERLAP_MAX_HORI, sinon $\frac{l}{\text{FACTOR\_SEEK\_ORIGINAL}}$;
- La hauteur de cette zone, $h - v_b uff -$ MIN_SEARCH_SPACE ;

Dans le cas vertical,

- La position du coin supérieur gauche, notée (x_{slave}, y_{slave}), est donnée par :
  - $x_{slave} =$ MIN_SEARCH_SPACE ;
  - $y_{slave} =$ MIN_SEARCH_SPACE ;
- La largeur de cette zone, $l - h_b uff -$ MIN_SEARCH_SPACE ;
- La hauteur de cette zone, si OVERLAP_MAX_VERT$\neq -1$, alors $=$ OVERLAP_MAX_VERT, sinon $\frac{h}{\text{FACTOR\_SEEK\_ORIGINAL}}$;

Illustration : Pour illustrer la sélection de la zone de recherche, reprenons l'image esclave (de résolution identique à l'image maître c'est-à-dire 768 sur 573) de la figure 3.19 et observons quelle zone est désignée. La figure 3.21 illustre le choix de limite de la zone choisie. Les valeurs des paramètres sont identiques à celles de l'étape précédente.

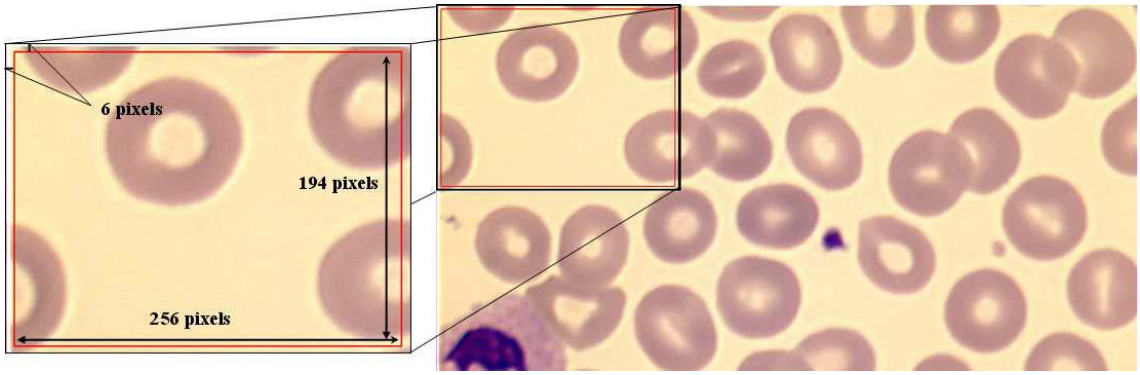

FIG. 3.21 – Sélection de la zone de recherche dans l'image esclave.

L'étape de coregistration

La troisième étape va effectuer la comparaison entre le vecteur étalon trouvé au point 3.3.2 et les différents vecteurs extraits (de taille v_{buff} ou b_{buff} selon que l'on coregistre horizontalement ou verticalement) au départ des pixels de la zone de recherche déterminée au point précédent. L'étape du point 2.3.3 n'a pas connu de modification mais nous reprenons tout de même le développement pour permettre au lecteur de suivre tout le processus en entier.

On obtient donc :
Pour une coregistration horizontale,

1. Pour les $l_{slave} \times h_{slave}$ colonnes t_i de taille v_{buff} de cette zone, l'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{v_{buff}} t_{ij}^2} \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

   et il calcule aussi la norme du vecteur étalon v_e et dont la norme est $\|v_e\|$;

2. un indice de comparaison c_i relatif à la colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{v_{buff}} |ve_j - t_{ij} \times \frac{\|ve\|}{\|t_i\|}| \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

3. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{l_{slave} \times h_{slave} - 1})$$

   Soit v_sol, le tableau de pixels dont les pixels sont les plus semblables à ceux du vecteur étalon. Ce tableau commence au pixel de coordonnées (x_{sol}, y_{sol}) et a, pour rappel, une longueur de v_{buff}.

Et pour une coregistration verticale, la méthode est la même mais on parlera de lignes et non de colonnes (ce qui implique que la longueur des tableaux de pixels manipulés est h_{buff} et non v_{buff}) :

1. Pour les $l_{slave} \times h_{slave}$ lignes t_i de taille h_{buff} de cette zone, l'algorithme calcule la norme de chacun des tableaux de pixels :

$$\|t_i\| = \sqrt{\sum_{j=0}^{h_{buff}} t_{ij}^2} \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

   et il calcule aussi la norme du vecteur étalon v_e et dont la norme est $\|v_e\|$;

2. un indice de comparaison c_i relatif à la colonne d'indice i est alors obtenu par :

$$c_i = \sum_{j=0}^{h_{buff}} |ve_j - t_{ij} \times \frac{\|t_i\|}{\|ve\|}| \qquad \forall \;\; 0 \leq i < l_{slave} * h_{slave}$$

3. Parmi tous les indices de comparaison calculés, on choisit le plus petit :

$$\min(c_0, c_1, ..., c_{l_{slave} \times h_{slave} - 1})$$

Soit v_{sol}, le tableau de pixels dont les pixels sont les plus semblables à ceux du vecteur étalon. Ce tableau commence au pixel de coordonnées (x_{sol}, y_{sol}) et a, pour rappel, une longueur de h_{buff}.

Modification de l'étape de positionnement relatif

Comme expliqué au point 2.3.4, cette quatrième et dernière étape vise à trouver le décalage à appliquer à l'image esclave pour la positionner dans l'image maître et cela à partir de la position de cette colonne et celle du vecteur étalon.

Pour cela, l'algorithme utilise les opérations suivantes :

- pour le cas horizontal :
  - $x = -x_{sol}$
  - $y = y_{ve} - y_{sol}$;
- pour le cas vertical :
  - $x = x_{ve} - x_{sol}$
  - $y = -y_{sol}$;

Illustration : Pour illustrer cette étape de positionnement, reprenons le cas des illustrations précédentes. Rappelons que les images ont une résolution de 768 pixels sur 573. La figure 3.22 montre la coregistration des deux images.

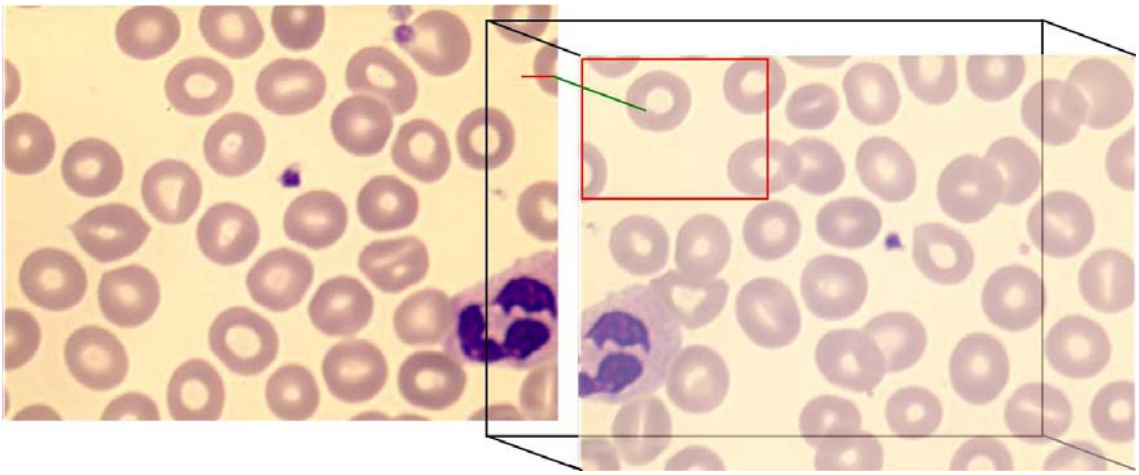

FIG. 3.22 – Positionnement relatif de l'image esclave par rapport à l'image maître

3.3.3 Modification de la gestion des requêtes de coregistration

Une autre modification apportée au serveur porte sur la manière dont le serveur gère les paramètres de coregistration. Auparavant, le serveur ne coregistrait qu'un seul type d'images. Lorsqu'il démarrait, il récupérait des paramètres dans un fichier de configuration.

Les paramètres de l'ancienne version étaient composés de sections pour :

- la gestion du cache du serveur d'images ;
- l'encodeur JPEG2000 ;
- le dossier où les images sources sont stockées ;
- la coregistration des images.

Chacune de ces sections contenait un certain nombre d'entrées correspondant à un paramètre. L'ensemble des paramètres était récupéré au démarrage et leurs valeurs étaient conservées dans l'application Java. Ce choix n'était plus adapté à la nouvelle utilisation du serveur de coregistration. Étant donné que celui-ci doit, à présent, être disposé à accepter des images de formats différents, il n'est plus concevable de modifier le fichier de configuration en adaptant les paramètres à chaque fois que l'on veut coregistrer des images sources de formats différents.

Deux changements ont été apportés au fichier de configuration. Le premier porte sur son contenu, le second sur le moment auquel les informations sont récupérées

Ainsi, le nouveau fichier de configuration contient les mêmes paramètres mais la gestion de ceux-ci est différente :

- la gestion du cache du serveur d'images ;
- l'encodeur JPEG2000 ;
- le dossier où les images sources sont stockées ;
- des paramètres de coregistration d'images **pour chacun des formats**.

On insistera sur les paramètres de coregistration qui sont dupliqués autant de fois qu'il existe de formats d'images sources.

Seules les sections de gestion du cache, de l'encodeur JPEG2000 et du dossier de stockage des images sources sont extraites au démarrage de l'application. La section concernant la coregistration n'est plus unique, il y a autant de sections qu'il y aura de formats d'images à coregistrer. De plus, elles ne sont récupérées que lorsqu'une demande de coregistration parvient au serveur. Ainsi, lorsqu'un client se connecte au serveur, il lui fournit l'ensemble d'informations que sont :

- l'adresse IP du serveur ;
- le numéro du port ouvert sur le serveur ;
- le nom de la section de paramètres de coregistration (c'est-à-dire le nom de l'ensemble des paramètres de coregistration relatif à un format d'images sources) ;
- le nom de l'utilisateur et le nom du dossier contenant les dossiers d'images sources ;
- le nom commun à toutes les images sources (qui est aussi le nom du dossier contenant ces images sources et le nom de la méga-image une fois celle-ci créée) ;
- le nombre d'images sources composant la largeur de la méga-image ;
- le nombre d'images sources composant la hauteur de la méga-image ;
- un coefficient d'homogénéisation, qui permet au serveur d'effectuer une homogénéi-

sation sur la méga-image créée.

Ces modifications apportent deux avantages :

- L'apparition de sections de paramètres de coregistration identifiables permet d'accepter différents formats d'images et "standardise" le serveur de coregistration.
- Le déplacement du moment de la récupération des paramètres de coregistration (lors de la réception d'une demande de coregistration et non plus au démarrage du serveur), apporte plus de souplesse au serveur. Il est, à présent, possible de modifier les paramètres de coregistration ou même d'ajouter une section entière sans devoir redémarrer le serveur. Ce qui rejoint la politique de client-serveur choisie.

3.4 Caractérisation des globules blancs

Cet objectif est à placer à la suite des travaux de Cédric Peeters (cfr [Pee07]). Ceux-ci fournissaient une application capable de visualiser des grands champs et d'en extraire des globules blancs. Il était donc possible de repérer les globules blancs et de les exposer sous forme de galerie. Par la suite, il m'a été demandé de travailler sur ce système afin de l'équiper de modules permettant de nommer et de différencier les globules blancs. Cet objectif a regroupé alors trois orientations :

- la première portant sur la classification elle-même, c'est-à-dire sur l'extraction de caractéristiques de globules blancs et de leur interprétation. Il s'agissait de dresser un ensemble de règles basées sur les caractéristiques extraites, ces règles permettant de déterminer les types de globules blancs observés.
- la seconde portant sur le système en lui-même, c'est-à-dire sur le caractère modulaire de l'application. En effet, il fallait avancer vers un système permettant de travailler n'importe quel type d'images et d'ajouter d'autres règles d'interprétation de caractéristiques afin de pouvoir analyser des images de prélèvements de moelle par exemple.
- la troisième portant sur l'analyse de la démarche des laborantins dans leur classification. Extraire les critères utilisés par l'humain et tenter de faire un parallèle avec des critères mathématiques manipulables par l'ordinateur.

3.4.1 Synthèse des caractéristiques à extraire

Sur base de la présentation des types des globules blancs développée au point 2.5.4, plusieurs caractéristiques peuvent être extraites par un traitement des images.

Taille de la cellule : La taille de la cellule permet de différencier certains globules blancs. En effet, leur taille varie de 7 à 40 μm, il est donc possible de créer des classes sur base de la taille de la cellule. L'unité de mesure pour caractériser la taille peut être le nombre de pixels. Il n'y a pas encore de mesures extraites. Il reste donc à trouver une méthode permettant de mettre en évidence cette caractéristique.

Durant les travaux préliminaires à ce mémoire, quelques méthodes de détection de contours ont été approchées pour tenter de mettre en évidence la cellule d'intérêt (voir partie 2.4.1). La figure 3.23 montre plusieurs globules de taille différente.

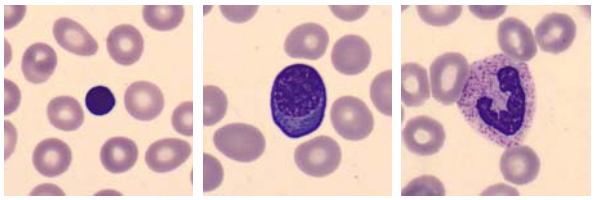

FIG. 3.23 – Illustration des différentes tailles de lymphocytes

Taille du noyau : La taille du noyau, au même titre que la taille de la cellule, a beaucoup d'intérêt. Les noyaux ne sont pas identiques d'une classe à l'autre. Elle peut se mesurer aussi au nombre de pixels composants le noyau dans l'image. Cette caractéristique a déjà été abordée précédemment par C. Peeters. Sa valeur figure dans le fichier Excel recensant les caractéristiques extraites. La technique est basée sur la binarisation de l'image et l'extraction par chaînage des pixels mis en évidence. La figure 3.24 montre différentes tailles de noyau de globule blanc.

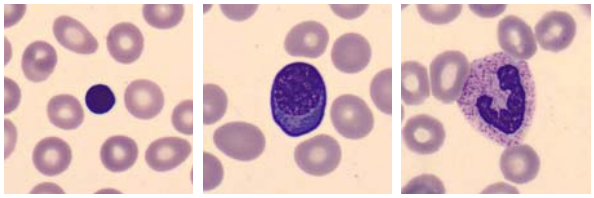

FIG. 3.24 – Illustration des différentes tailles de noyau

Forme de la cellule : La forme de la cellule participe à la différenciation des types. Cette caractéristique peut être liée à celle relative à la taille de la cellule. L'analyse de la forme pourra donc être dérivée d'une analyse des contours de la cellule (qui n'a pas encore été abordée et le sera dans ce mémoire). La figure 3.25 propose 3 formes différentes de cellule de globule blanc.

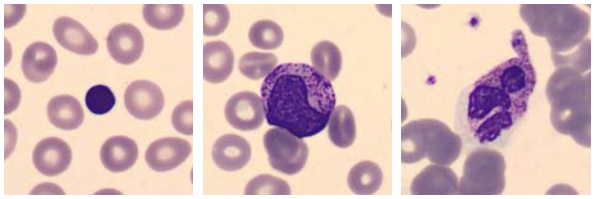

FIG. 3.25 – Illustration des différentes formes de cellules

Régularité du noyau : Tout comme la forme de la cellule, la forme du noyau permet de classifier. La partie 2.5.4 décrit certains noyaux de globules blancs au travers d'un certain nombre de lobes reliés entre eux. Cette caractéristique peut être calculée sur base d'une variation des bords du noyau. Elle a fait l'objet de travaux dans la suite de ce mémoire. La figure 3.26 illustre les diversités de formes des noyaux.

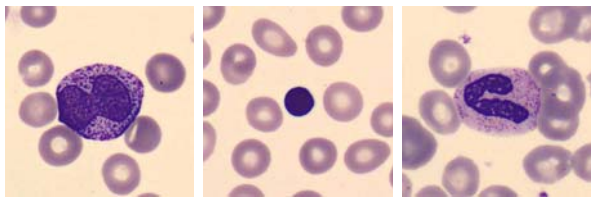

FIG. 3.26 – Illustration des différentes formes de noyau

Ratio nucléo-cytoplasmique : Ce rapport est la comparaison de la taille du noyau et de la taille de la cellule. Il pourra donc être mesuré dès que ces deux valeurs auront pu être évaluées. La figure illustre la forte variabilité du ratio nucléo-cytoplasmique.

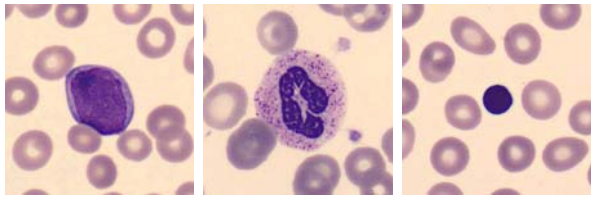

FIG. 3.27 – Illustration des différents rapports Taille du noyau/Taille de cellule

Couleur du noyau et de la cellule : la couleur du noyau et de la cellule peuvent différer (figure 3.28) selon les classes de globules blancs (voir la section 2.5.4). Par manque de temps, cette caractéristique jugée moins prioritaire, n'a pas encore fait l'objet de recherches et n'a pas été abordée dans mes travaux.

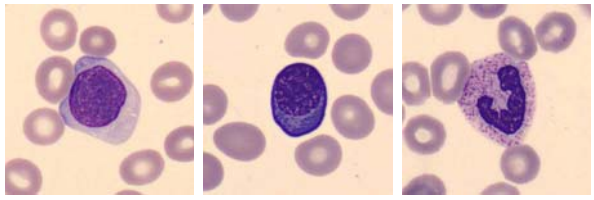

FIG. 3.28 – Illustration des différentes couleurs des cellules et du noyau

Présence de nucléoles : Certains noyaux présentent des nucléoles, c'est-à-dire des zones corpusculaires. Il est intéressant de les dénombrer. Aucun travail n'a été réalisé dans cette optique et aucune piste n'a encore été trouvée pour mettre en évidence ces corpuscules. La figure 3.29 montre la variabilité du nombre de nucléoles dans le noyau.

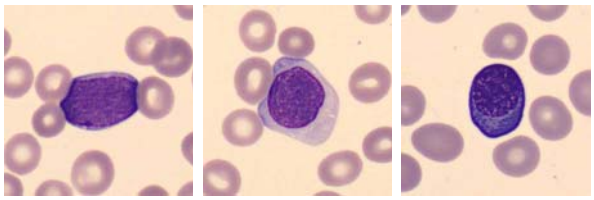

FIG. 3.29 – Illustration des différentes cellules contenant pas, peu ou beaucoup de nucléoles

Dans la suite du développement de Cédric Peeters, il s'agissait de compléter les fichiers Microsoft Office Excel (.xls) en y ajoutant de nouvelles valeurs issues du traitement de l'image.

Plusieurs pistes ont été suivies :

– l'analyse de contours de la cellule et non plus simplement du contour du noyau.
– l'ajout de caractéristiques calculées permettant de caractériser la régularité du noyau.

3.4.2 Traitement des contours

Les résultats obtenus dans les travaux précédents se basaient sur la localisation du noyau. Or, il est important de considérer l'ensemble du globule blanc pour en extraire certains critères (taille de la cellule, rapport entre la taille de la cellule et taille du noyau,...). Ainsi, nous avons exploré différents algorithmes de repérages des contours afin de mettre en évidence les contours de la cellule (et non plus simplement les contours du noyau).

Nous avons donc testé des méthodes de contours qui ont déjà été citées dans [Pee07] avec différentes applications (voir [PW03] et [PIAG08]). Reprenons ces méthodes et observons les résultats à la figure 3.30.

L'ensemble des résultats ne permettait pas d'utilisation directe. Cela s'explique par le fait que, pour pouvoir caractériser le contour du globule blanc, il était nécessaire d'obtenir un contour continu et d'un pixel d'épaisseur. Les résultats des filtres de Sobel et de Prewitt ne permettent pas de différencier les contours de globules rouges et les globules blancs.

Le filtre de Canny, à première vue, ne semble pas convenir. Mais l'application en ligne [PW03] proposait un filtre pour réduire l'épaisseur du contour à 1 pixel. Les résultats semblent assez bons pour la première cellule filtrée (cfr (a) et (e) figure 3.30). Mais ce résultat ne peut pas être généralisé à chaque type de globules blancs. En effet, si on observe la figure (j) de la figure 3.30, l'extraction de contours est perturbée par les granulations contenues dans le cytoplasme.

Le temps nous a manqué pour poursuivre les recherches sur cet objectif moins prioritaire. Les méthodes qui ont été testées doivent être écartées pour une utilisation stricte[8],

[8]On entend par utilisation stricte, une utilisation de la méthode sans pré-traitement ou post-traitement

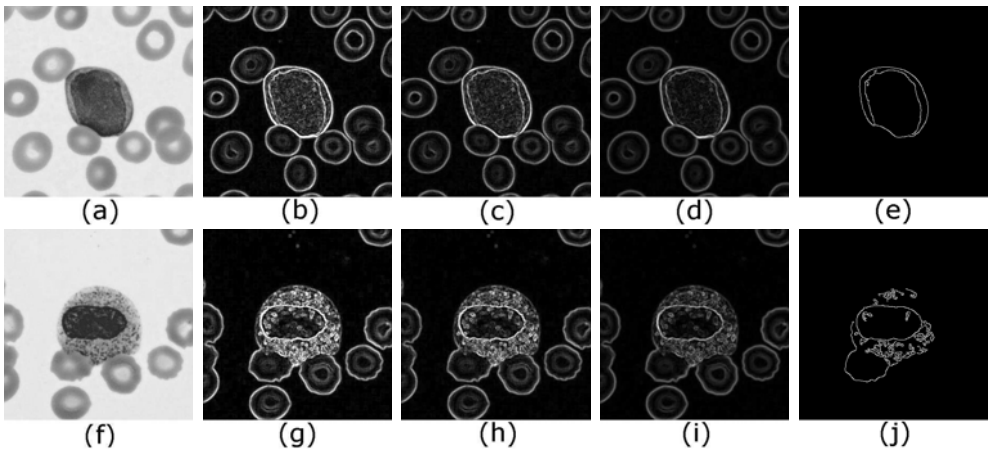

FIG. 3.30 – Application des méthodes de Sobel (b et g), Prewitt (c et h), Canny (d et i), Canny avec affinage des contours (e et j) sur des images de globules blancs (a et f).

mais la méthode de Canny pourrait être une piste intéressante. Il sera nécessaire d'utiliser des techniques complémentaires pour affiner les contours détectés par exemple.

3.4.3 Caractéristiques ajoutées

Pour rappel, l'application développée par C. Peeters extrayait des caractéristiques telles que le nombre de pixels constituant le noyau des globules et la taille de la zone rectangulaire dans laquelle se trouve le noyau. Comme nous l'avons vu dans la section 2.5.4, la forme du noyau, sa régularité, est un point de choix dans la classification. Il est intéressant de mettre en place des valeurs numériques permettant de caractériser la forme du noyau.

L'idée de la méthode est basée sur le calcul du centre de la cellule et d'un coefficient de variation du contour. Ainsi pour chacun des globules blancs repérés, on recense les pixels composant le contour et les pixels internes. A partir des pixels formant le contour, il est possible de trouver le centre de la cellule via le calcul de la moyenne des coordonnées. Une fois le centre trouvé, on peut obtenir la distance le séparant de chacun des pixels du contour en utilisant la distance euclidienne. Il ne reste plus qu'à calculer un taux de variation. Mes travaux ont utilisé la méthode basée sur le rapport entre l'écart-type et la distance au noyau, $\frac{\sigma}{\mu}$.

Nous l'avons vu, nous avons la distance de tous les points de contour par rapport au centre. La moyenne μ de ces distances, aussi notée $E[x]$, peut être facilement obtenue, de même que la moyenne des distance au carré, $E[X^2]$. De ces calculs, la variance est (qui est l'écart-type au carré, σ^2) :

$$\sigma^2 = E[X^2] - E[X]^2$$

Nous n'avons pas pu terminer cette idée et vérifier si ce critère de variation était utilisable.

Malgré tout, des caractéristiques ont pu être ajoutées aux fichiers MS Excel (colonnes I et J de l'image (a) de la figure 3.31). Ainsi, lorsqu'un globule est détecté, une ligne est

ajoutée contenant les caractéristiques de l'image. Une feuille est aussi ajoutée pour stocker les coordonnées (en x et en y) de chacun des pixels composant l'élément détecté ((b) de la figure 3.31). Pour chaque pixel, on précise aussi s'il est un contour du noyau ou un pixel intérieur et sa distance par rapport au centre du globule (respectivement colonnes C et D de l'image (b) de la figure 3.31).

Le stockage de toutes ces informations permet de traiter le fichier MS Excel directement si celui-ci existe (à la place de retraiter toute l'image) (voir la modification de l'architecture pour la caractérisation).

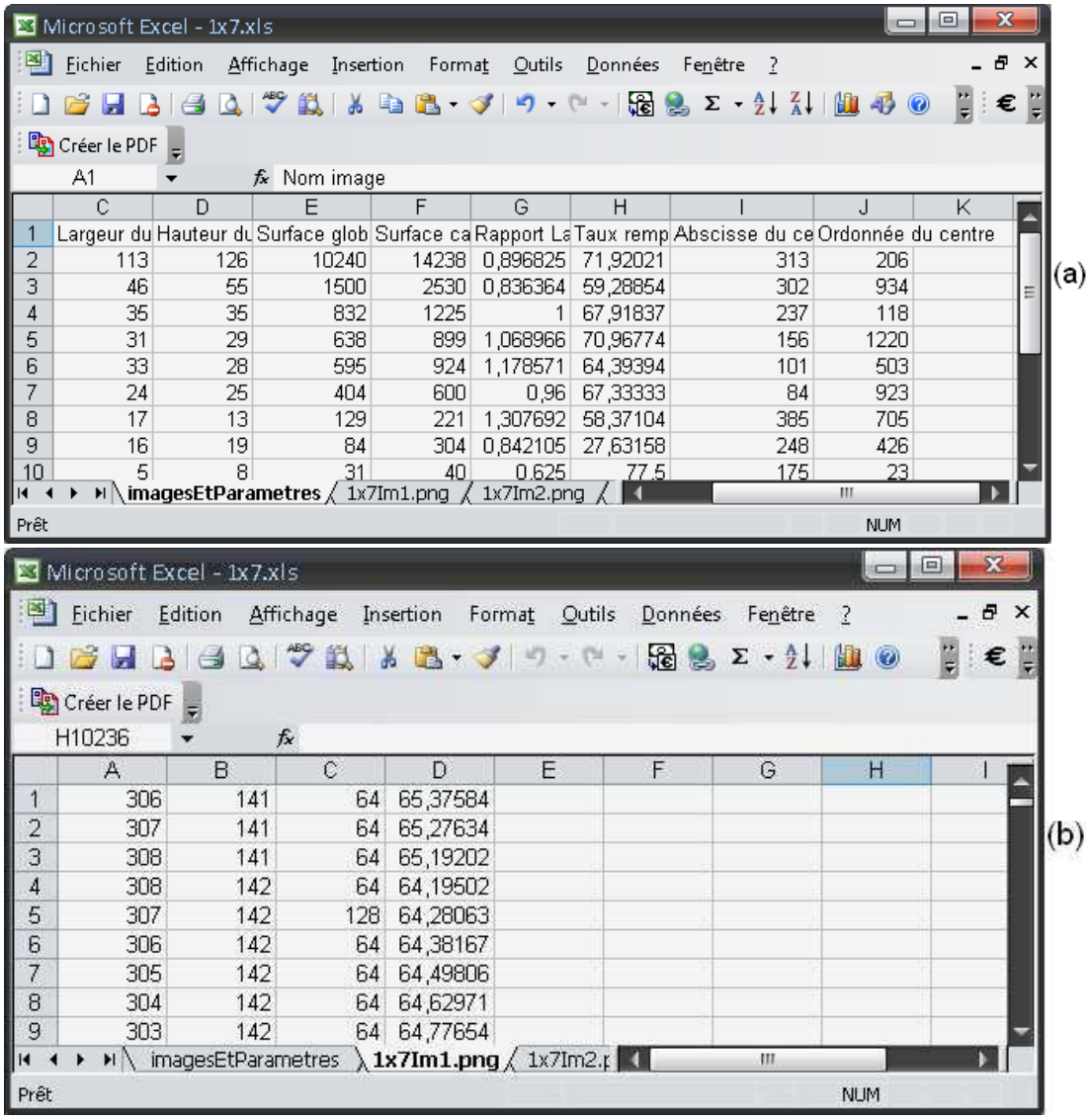

FIG. 3.31 – Capture d'écran des fichiers MS Excel : (a) Critères des globules blancs, (b) Aperçu d'une feuille reprenant les pixels composant un globule.

3.4.4 Modification de l'application de caractérisation

L'architecture du module de caractérisation a été légèrement modifiée.

La première ne vise qu'à accélérer le temps de traitement. En effet, précédemment, le programme traitait l'image à chaque demande. Il créait donc le fichier MS Excel à chaque fois. A présent, le programme débute par la vérification de la présence du fichier Excel et si celui-ci n'existe pas, effectue le traitement de l'image. Cette solution permet d'accélérer fortement la caractérisation. C'est dans cette optique que chacun des pixels du globule est encodé lors du seul et unique traitement.

Une autre idée a été de déléguer les calculs (tels que la superficie du globule, ou encore la superficie du rectangle entourant ce globule) au fichier MS Excel. En effet, l'API jxl permet d'insérer directement des formules Excel.

Illustration : Le code qui suit illustre l'introduction de formules au sein des fichiers MS Excel. Il insère dans la case située dans la colonne 4 et à la ligne 3, la formule "E1+E2" (c'est-à-dire la somme du contenu de la case E1 et E2).

```
WritableSheet s = workbook.createSheet("Sheet", 0);
/**Créé une formule à la ligne 4 et la colonne 3, contenant la
   somme de la case E1 et E2*/
Formula f = new Formula(4,3, "E1+E2");
s.addCell(f);
```

Cette solution permettrait de gagner du temps de calcul, le programme pourrait se consacrer à d'autres choses pendant que le fichier MS Excel effectue les formules. Elle est seulement limitée par le risque de formules cycliques. Ces dernières sont des formules faisant appel à des cases qui ont besoin du résultat de ces mêmes formules pour effectuer le calcul.

Cette idée n'a pas apporté les effets attendus. Il semble que la récupération des formules ne soit pas tout à fait au point. En effet, elle retournait une valeur incohérente et identique quelque soit la formule.

Nous avons cependant mis en place une ébauche de classification. En effet, deux avancées sont visibles dans la figure 3.32.

La première est l'apparition de noms sous les images de la galerie (n°1 sur la figure 3.32). Les noms sont issus du traitement des caractéristiques déjà extraites. L'application de filtre de binarisation repère aussi les plaquettes qui sont plus petites que les globules blancs. Nous avons donc mis en place un premier niveau d'interprétation des caractéristiques, plus particulièrement le nombre de pixels composant un "globule blanc" (ou plaquette) détecté. Nous avons donc pu déjà mettre en évidence ces plaquettes (thrombocytes) via la définition d'une borne entre la taille des globules et des plaquettes.

La seconde est l'apparition du niveau de l'objectif utilisé lors de la création de la méga-image (n°2 sur sur la figure 3.32). Il est en effet important de prendre en compte la taille de l'objectif lors de la caractérisation. Il est évident que les cellules repérées sur une image capturée avec un objectif 40x comporteront moins de pixels que les cellules repérées sur une image d'objectif 100x. Dans cette optique, nous avons inséré dans l'interface d'ouverture

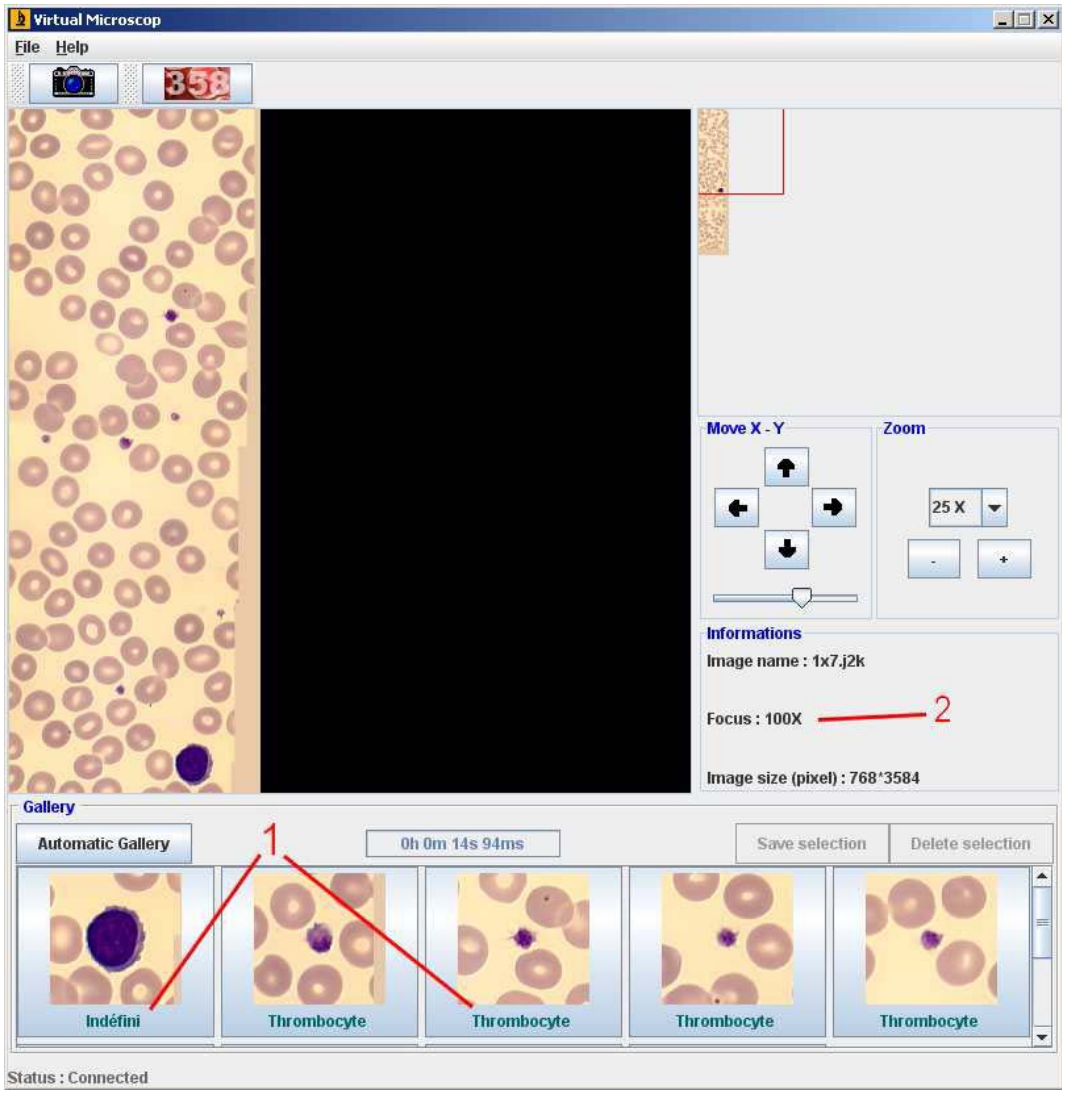

FIG. 3.32 – Interface du nouveau visualisateur (avec certains composants classifiés)

de méga-image une option permettant à l'utilisateur de spécifier le type d'objectif utilisé (voir figure 3.33).

De manière plus générale, l'application a été modifiée pour la rendre plus modulaire. En effet, si la caractérisation de globules blancs donne de bons résultats, pourquoi ne pas imaginer faire le même raisonnement pour d'autres types d'images (échantillon de moelle, etc). Dans cette optique, l'application a été modifiée pour permettre facilement l'ajout de modules Java offrant des règles de caractérisation de contenu d'autres images. Cette tendance n'a pas encore pu être appliquée sur toute l'application et devra être poursuivie dans les développements futurs.

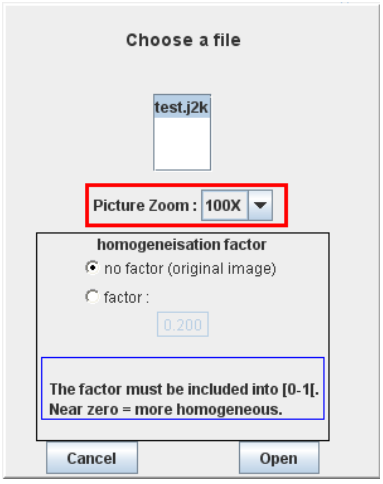

FIG. 3.33 – Définition de l'objectif à l'ouverture de la méga-image.

3.5 Extraction et envoi des résultats d'analyses

Dans la continuité du travail de C. Peeters (cfr [Pee07]), un des objectifs du laboratoire était de visualiser et partager des "archives" d'analyses stockées au format .mdb(Microsoft Office Access).

Comme décrit dans la partie 1.5, le laboratoire s'est équipé de ***CellaVision***, un outil comprenant un appareil de capture numériques de lames et un logiciel de traitement (voir figure 3.34 et [Cel08]) qui permet d'analyser automatiquement les lames reçues par le laboratoire. Les résultats sont visualisables sous la forme d'une galerie d'images de globules blancs et contenant les informations relatives à l'analyse et au patient.

Une fois l'analyse effectuée, les images "témoins" sont conservées dans un premier temps sous forme de base de données propre à CellaVision. Les laborantins peuvent alors valider les résultats pour archiver les images. L'archivage se réalise au moyen de base de données .mdb, de MS Office Access. Une fois archivées, les images sont inutilisables. Car, premièrement, l'ordinateur sur lequel fonctionne CellaVision ne dispose pas de logiciels permettant de lire des fichiers .mdb et, deuxièmement, même en supposant qu'un utilisateur fasse une copie pour lire le fichier .mdb sur une machine équipée, la lecture de ce fichier sera difficile au vu du nombre de tables qu'il contient[9].

Jusqu'à présent, le laboratoire conservait les bases de données sur des disques durs externes. Ceux-ci se font de plus en plus nombreux, et représentent une masse importante de données à gérer. Et comme nous le disions plus haut, la manière et l'endroit de stockage des informations les rendaient inutiles.

Il est donc intéressant de pouvoir visualiser le contenu des .mdb mais aussi de le partager en stockant les informations utiles[10] sur des serveurs dédiés, partagés. Serveurs pouvant

[9]Il contient 73 tables. La lecture sera encore plus compliquée si le nombre d'analyses que le fichier contient est élevé.

[10]Les informations utiles sont des informations comme les images des globules détectés, les informations

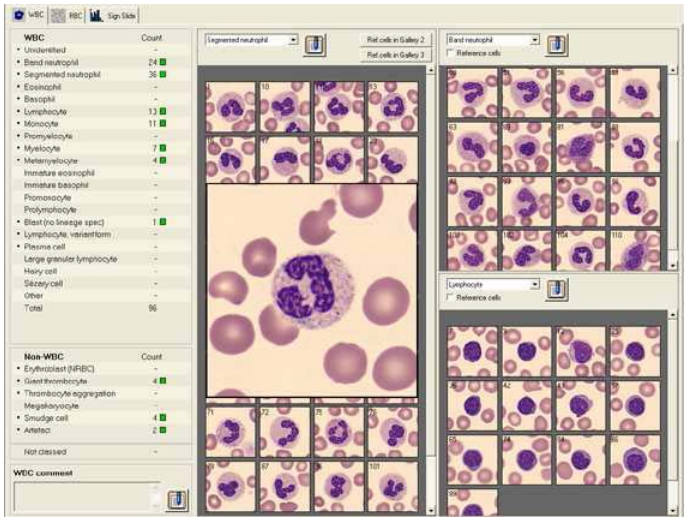

FIG. 3.34 – Interface de la galerie CellaVision

être accessibles à partir d'autres stations de travail que celles du laboratoire.

L'idée avait déjà été mise en place dans les travaux de C. Peeters. Son application permettait déjà d'extraire des fichiers .mdb (Microsoft Office Access) ainsi que de formater une bonne partie de l'ensemble des données en des objets DICOM pouvant être envoyés sur le serveur distant.

Il restait cependant du travail à fournir. Il fallait compléter l'application pour que toutes les informations utiles disponibles dans CellaVision soient utilisées dans les champs visibles à la visualisation ou dans les champs DICOM des images envoyées.

L'entreprise produisant CellaVision a complété son logiciel pour qu'il récupère les données de patients ou analyses et surtout qu'il les insère dans les bases de données d'archives Microsoft Access(.mdb). Suite à cet update, l'application a pu être complétée et testée.

Cependant, suite aux tests, des problèmes se sont révélés. Le problème majeur était que l'application n'acceptait pas les fichiers .mdb contenant plus d'une analyse. Ce qui était assez contraignant pour les laborantins qui traitent jusqu'à 100 analyses par jour, et donc par fichier.

Le second problème provenait des requêtes d'extraction des images. Celles-ci ne sélectionnaient pas toutes les images de l'archive avec pour conséquence que certaines images n'étaient :

– soit pas envoyées au serveur (si on utilise le service) ;
– soit pas visibles dans le visualisateur.

Les requêtes étaient soumises à des conditions trop restrictives. L'extraction ne repérait qu'une petite partie de l'ensemble des images. Par exemple, une base de données devant contenir plus de 1000 images n'en extrayait que 36.

---

de classification données par l'analyse CellaVision mais aussi les informations concernant le patient.

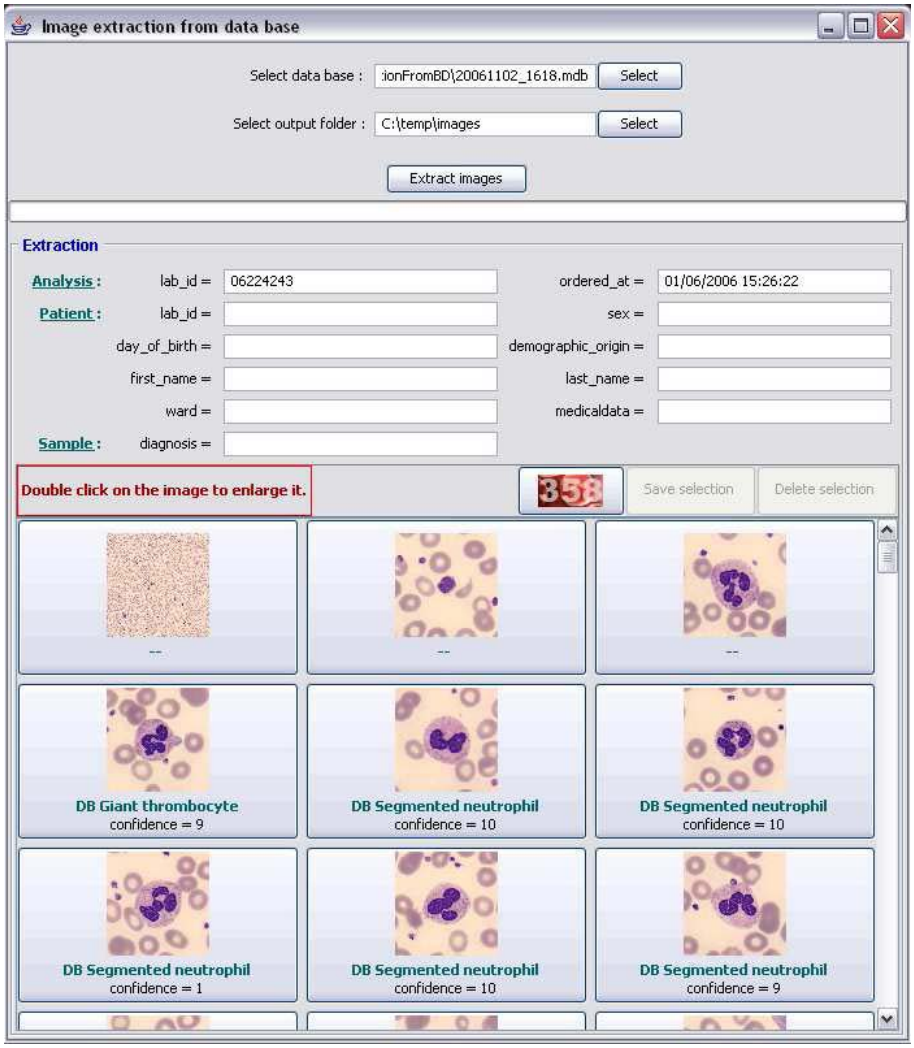

FIG. 3.35 – Interface de la galerie extraite développée par C. Peeters

3.5.1 Extraction de plusieurs analyses

Développons le premier problème rencontré. Les jeux de tests qui avaient été fournis à C. Peeters ne contenaient qu'une seule analyse. Pour pouvoir tester la correction apportée (compléter les champs avec les valeurs adéquates), de nouveaux jeux de tests m'ont été fournis. Ceux-ci contenaient plusieurs analyses et ont donc permis de montrer le manquement.

Afin de corriger ce défaut, il a fallu repenser la méthode d'extraction. La précédente ne permettait pas de créer les liens entre les informations de l'analyse et les images la constituant.

Modification de l'application "orienté visualisation" L'application construite pour la visualisation du contenu de la base de données a dû être adaptée afin de pouvoir traiter les requêtes de l'interface. La version précédente extrayait l'ensemble des images et des informations et insérait le tout dans une liste. Cette conception ne permettait pas de différencier les différentes analyses extraites. Nous avons modifié la méthode en créant un objet "Analyse", sous-classe de "java.util.Vector". Cette classe permet de structurer les données d'une analyse. Elle référence une "AnalyseInfos" contenant les informations communes à toutes les images. Ces informations sont :

- le nom et prénom du patient ;
- la date de naissance et le genre du patient ;
- l'origine démographique du patient ;
- le numéro identifiant de l'analyse et du patient au sein du laboratoire ;
- la date de demande d'analyse,
- des données médicales et des notes pour le diagnostic.

Par la suite, pour chaque image de globule blanc extraite, l'application crée un objet "ImageAndData" étendant la classe "java.util.Vector". Cet objet stockera l'ensemble des classifications attribuées par la traitement de CellaVision (sous forme de vecteur de "Classification") et l'adresse du fichier de l'image extraite. Une fois créé, le "ImageAndData" est ajouté au vecteur de l'analyse correspondante.

Une fois le contenu de l'analyse rassemblé, l'objet "Analyse" est stocké dans une Hashtable (java.util.Hashtable) qui sera donnée au composant intéressé par le contenu de la base de données traitée.

Modification de l'application "orienté service" L'orientation service a aussi dû être modifiée afin de pouvoir gérer les archives multi-analyses. En effet, les modifications sont du même ordre que pour l'orientation visualisation à laquelle sont ajoutées des modifications liées à l'interface et à l'organisation du processus interne.

La nouvelle architecture est multi-thread. Ce choix est motivé par la grande quantité d'informations à traiter et la nécessité de ne pas surcharger la machine virtuelle Java. De cette manière, deux threads ont été mis en place. L'un se chargeant d'extraire les informations, de créer les liens avec les images extraites et de formater le tout dans des objets DICOM qui sont alors stockés dans une liste (d'objets DICOM). Un second thread, démarrant en même temps, est chargé d'interroger cette liste d'objets DICOM, et si celle-ci n'est pas vide, de récupérer un objet pour l'envoyer au serveur partagé distant.

3.5.2 Correction des requêtes

Le second problème cité plus haut était la saisie des images dans la base de données. Les requêtes utilisées n'avaient, par manque de temps et de disponibilité, pas été validées. Les développements qui ont suivi se basaient sur la validation de l'existant. Lors du déploiement et de l'analyse des résultats, le nombre d'images envoyées ne correspondait pas, il a fallu modifier la requête.

Ainsi la requête ci-dessous ne permettait pas de récolter toutes les images. Les conditions de la clause "WHERE" ne sont pas correctes. Elles faussent le nombre d'images sélectionnées.

```
SELECT images.id, classname.name, classification.confidence,
FROM (((classification INNER JOIN classname ON classification.
    classname = classname.id)
INNER JOIN images ON classification.point = images.id) INNER JOIN
     run ON images.run = run.id)
INNER JOIN analysis ON run.analysis = analysis.id
WHERE (((classification.confidence) Is Not Null)
AND ((classification.correct) Is Null))
```

Ces conditions ne sont pas nécessaires. Il a donc suffit de les supprimer. Les champs récupérés ont aussi été modifiés pour répondre aux besoins stricts de l'application.

```
SELECT images.id, images.jpg, analysis.lab_id
FROM (images INNER JOIN run ON images.run = run.id)
INNER JOIN analysis ON run.analysis = analysis.id
```

3.5.3 Interface développée

L'interface proposée par C. Peeters (voir figure 3.35) a été conservée et complétée par quelques fonctionnalités. L'application de C. Peeters était, nous l'avons vu, dérivée en deux orientations : l'une orientée IHM, l'autre orientée service DICOM.

Les deux applications étaient indépendantes et complémentaires : l'interface permettant de visualiser les images ne permettait pas l'envoi DICOM, et le service ne permettait pas de visionner les images.

Les développements réalisés n'ont pas modifié cette optique. L'organisation de l'interface de visualisation du contenu de la base de données n'a pas été modifiée mais adaptée au contenu de la base de données. En effet, le temps manquait pour mettre au point une interface répondant parfaitement au contenu de la base de données. Ainsi, l'IHM précédemment développée ne permettait de visualiser qu'une seule analyse du fichier .mdb[11]. L'interface a donc été modifiée afin de permettre la navigation entre les différentes analyses extraites. Ainsi deux boutons apparaissent dans l'interface (figure 3.36) permettant de parcourir les analyses extraites.

Cette fonctionnalité a nécessité des modifications dans l'architecture de l'application de visualisation. En effet, l'ancienne architecture fournissait à l'interface une liste d'adresses d'images auxquelles étaient associées leur information. Il a fallu ajouter une couche pour différencier les différentes analyses.

Ainsi l'interface reçoit une hashtable (java.util.Hashtable) correspondant à une liste d'analyses. L'utilisation des nouveaux boutons de l'interface entraîne l'appel à la hashtable pour récupérer l'analyse suivante ou précédente (en fonction du bouton sélectionné).

[11] Ce qui n'aurait pu se réaliser étant donné que l'extraction, elle-même, ne retirait qu'une seule analyse, voir le point 3.5.1

L'interface chargera alors les informations concernant l'analyse (nom et prénom du patient, identifiant de l'analyse,...) ainsi que toutes les images qu'elle référence.

Une autre modification de l'interface a été de permettre l'envoi des images sélectionnées. Dans ce but, un bouton "Send Selection" a été ajouté aux boutons de la galerie. L'utilisateur peut, grâce à cette fonctionnalité, parcourir une analyse, sélectionner des aperçus et les envoyer sur un serveur DICOM distant.

Pour compléter cette fonctionnalité, deux boutons complémentaires ont été ajoutés. Ceux-ci permettent de sélectionner ou désélectionner toutes les images de l'analyse sélectionnée. Cela accélère la (dé)sélection pour l'utilisateur.

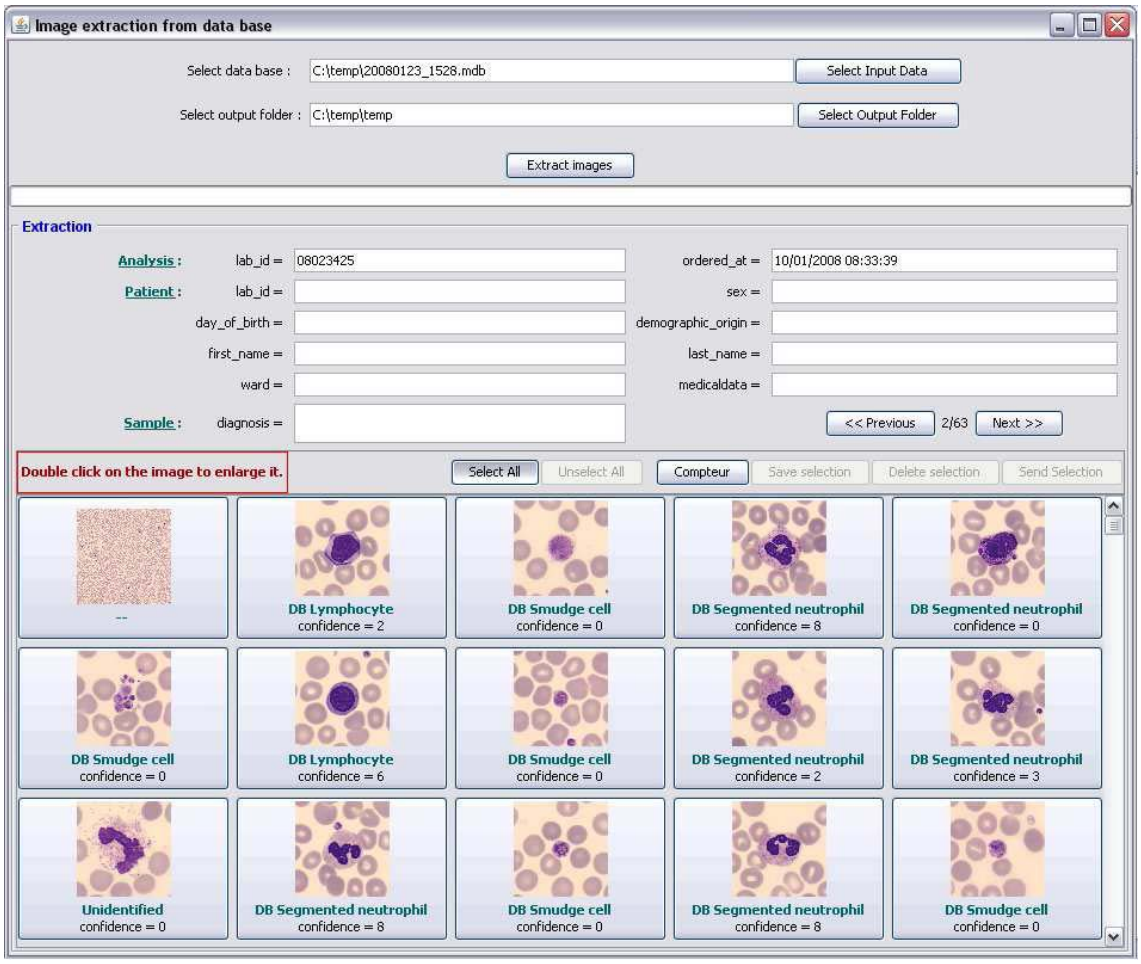

FIG. 3.36 – Interface permettant de visionner les analyses extraites.

Cette nouvelle interface offre, à présent, la possibilité à l'utilisateur de visualiser toutes les analyses de l'archive via les boutons "Next" et "Previous".

L'approche service de l'application a aussi été munie d'une petite interface. Celle-ci consiste en une icône apparaissant dans la barre des tâches. Un "clic" sur cette icône donne accès à un menu contenant les boutons suivants :

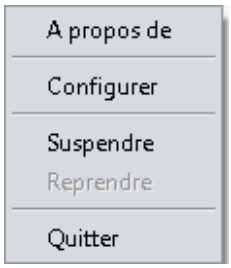

FIG. 3.37 – Menu disponible via la barre des tâches

La figure 3.37 est composé de différents boutons :
- "Quitter" permet de quitter l'application ;
- "Suspendre" permet de suspendre l'application, c'est-à-dire de mettre en pause les threads d'extraction et d'envoi ;
- "Reprendre" permet de redémarrer l'application, c'est-à-dire de relancer les threads d'extraction et d'envoi ;
- "Configuration" permet de faire apparaître une fenêtre de configuration. Celle-ci sera détaillée ci-dessous.

La fenêtre de configuration (figure 3.38) offre un accès à l'utilisateur pour modifier les variables de fonctionnement de l'application. Elle comprend donc de multiples champs :
- "Adresse DICOM" permet à l'utilisateur de choisir le serveur distant sur lequel seront envoyées les images (cfr le fichier "*aet.cfg*") ;
- "Temps entre les vérifications" permet de spécifier le temps entre chaque vérification de présence de base de données .mdb ;
- "Dossier Input" permet de spécifier le dossier dans lequel l'application va vérifier la présence d'archives à extraire ;
- "Dossier Output" permet de spécifier le dossier dans lequel l'utilisateur souhaite extraire les images extraites des archives ;

Le paramètre "Temps entre les vérifications" n'était pas présent dans l'application précédente. L'application de C. Peeters effectuait une vérification dans un dossier déterminé toutes les 10 secondes et vérifiait si ce dernier était vide. Nous avons donc introduit une variable permettant à l'utilisateur de configurer ce rythme de vérification. Ainsi si le laboratoire ne souhaite faire qu'un ou deux envois par jour, il peut configurer ce champ respectivement par 86400 secondes ou 43200 secondes.

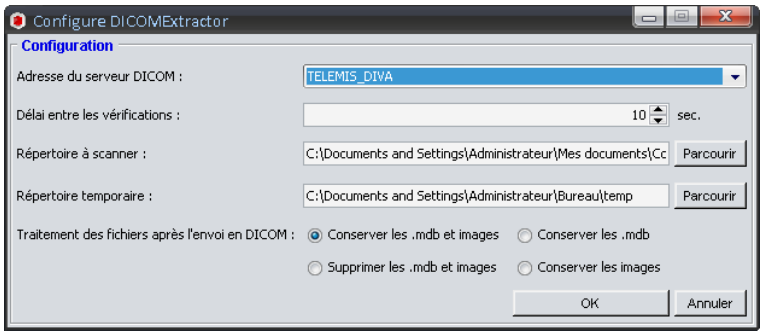

FIG. 3.38 – Interface de configuration du service d'extraction.

Une fois la configuration effectuée, le processus responsable de l'extraction va traiter les fichiers d'archives .mdb (Microsoft Office Access) et envoyer leur contenu sous forme d'objets DICOM sur le serveur distant.

Chapitre 4

Discussion

4.1 Introduction

Après les explications des résultats obtenus pour chacun des objectifs décrits dans 1.6, il est nécessaire de porter un regard critique sur ce qui a été fait et surtout de proposer des pistes pour la poursuite des recherches.

Le chapitre précédent a permis de mettre en évidence les résultats des travaux sur l'autofocus, sur la caractérisation des globules blancs, sur la coregistration indépendante de la résolution des images sources et sur l'extraction de contenu d'une base de données. Tous les résultats n'ont pas permis de solutionner les problèmes posés. Ainsi, sur ces 4 objectifs, un est arrivé à un blocage (l'autofocus défaillant), et les 3 autres ont donné des résultats satisfaisants mais nécessitant des améliorations et/ou des tests complémentaires.

Cette partie, intitulée "Discussion", va nous permettre de critiquer les résultats obtenus mais surtout de proposer des orientations et des idées pour l'amélioration des développements.

4.2 Autofocus

Le développement proposé au point 3.2 a permis de recenser l'ensemble des composants utiles à la mise en place d'une système de télémicroscopie efficace. Ainsi, nous connaissons à présent les liens entre chacun des composants du système, leur utilisation et leur interaction.

Il était bien nécessaire de clarifier l'ensemble de l'infrastructure pour comprendre les solutions proposées et les solutions envisageables.

Il a aussi présenté une solution et les résultats de la mise en place de celle-ci.

Cette solution n'a pas permis de résoudre le problème proprement dit. Le partage de l'axe Z entre l'autofocus et le moteur amovible empêche la mise en place de la solution imaginée. Mais elle a, par contre, permis de découvrir et clarifier la tâche et l'utilisation de chacun des composants de l'environnement du microscope. Ainsi, nous savons, à présent, que le moteur est utilisable et fonctionnel. Nous savons aussi que le moteur agit sur le même axe que l'autofocus ce qui empêche leur utilisation simultanée.

Forts de toutes ces informations, nous pouvons donc proposer d'autres pistes à explorer

L'autofocus, étant conflictuel avec le moteur de l'axe Z, deux solutions sont envisageables et sont basées sur l'utilisation exclusive d'un des deux composants :

- l'utilisation exclusive de l'autofocus est justement la source du problème qui nous occupe. Lorsque l'autofocus ne réussit pas, la platine se perd sur son axe Z. A moins de mettre à jour la version de l'autofocus pour qu'elle accepte les fonctions manquantes, cette solution est écartée d'elle-même.
- l'utilisation exclusive du moteur de l'axe Z. Cette solution consiste donc à utiliser ce moteur pour modifier l'axe Z dans le processus de mise au point.

Cette deuxième idée nécessite une application permettant de mesurer la netteté d'une image. L'autofocus actuel serait alors supprimé pour un autofocus software. Cette solution demandera vraisemblablement plus de temps à l'exécution, le calcul de netteté devant être calculé pour chacune des images capturées.

4.3 Coregistration d'images

La technique de coregistration développée dans la partie 3.3 offre des résultats équivalents à l'ancienne technique tout en rendant le serveur beaucoup plus indépendant et plus souple par rapport aux formats des images sources. Les travaux ont permis de mettre en place une nouvelle méthode qui offre maintenant plus de souplesse en tout point.

En effet, dorénavant, la coregistration donne des résultats pour des images sources de résolution bien différentes. Ce qui permet au laboratoire de pouvoir utiliser le serveur quelque soit la camera utilisée pour la capture. L'avantage est un gain de temps important car la coregistration manuelle nécessitait une personne durant toute la procédure de coregistration. Maintenant, il suffit de fournir les bons paramètres, de lancer le client de coregistration et d'attendre le résultat. L'utilisateur peut alors affecter ce temps d'attente à une autre activité. De plus, la coregistration automatique est plus rapide que la manuelle (dans l'hypothèse où le système dispose de paramètres corrects et précis).

La modification du serveur au niveau du moment de récupération des données fournit aussi un avantage supplémentaire non négligeable. Á présent, il n'est plus nécessaire de redémarrer le serveur lorsque l'on veut coregistrer des images sources de résolutions différentes. Le serveur récupère les valeurs après avoir reçu une demande de coregistration, ce qui lui permet de choisir dans le fichier de paramètres, les informations liées aux formats d'images sources à coregistrer.

Il est cependant encore nécessaire de tester l'application et surtout de vérifier l'influence de la méthode de correction des erreurs et de création de la méga-image mise en place par L. Zuyderhoff. Le temps nous a manqué pour effectuer une analyse complète de la méthode. Il semble en effet que la nouvelle méthode laisse apparaître un léger décalage lors des coregistrations. Ce décalage est beaucoup plus visible sur des images de "basses" résolutions (Ce décalage est plus visible sur les images de 768 sur 573 pixels). Ce problème pourrait provenir des indices de travail sur les tableaux. En effet, le traitement des tableaux demande de démarrer les index à 0, mais les utilisations des résultats considèrent peut-être que l'indice commence à 1. Cette considération influe alors sur la position de l'image esclave sur l'image maître. Ce type d'erreur peut survenir à deux endroits, soit au calcul du

décalage, soit à la création de la méga-image c'est-à-dire à l'étape qui consiste à parcourir toutes les images sources pour créer la méga-image.

Autre réflexion, le choix du tableau de pixels noirs lors de l'étape de sélection du vecteur étalon dans l'image maître. En effet, nous l'avons vu si on reprend le point 1 ou 2 du développement de la section 3.3.2, les tableaux buffer ($BufV$ ou $BufH$, utilisés ci-dessous avec l'appellation générique de $Buff$ et dont la longueur h_{buff} ou v_{buff} est appelée l_{buff}) sont remplis de la valeur 0. Or, si on reprend la fonction utilisée pour la norme, on s'aperçoit que celle-ci donnera 0 comme résultat.

En effet, le calcul de la norme pour le buffer (vertical ou horizontal) donnera :

$$\|Buff\| = \sqrt{\sum_{j=0}^{l_{buff}} 0^2} = 0$$

En continuant le développement, on s'aperçoit que tous les indices de comparaison c_i relatifs à la colonne d'indice i seront toujours égaux est 0. Ainsi on obtient :

$$c_i = \sum_{j=0}^{l_{buff}} |0 - t_{ij} \times \frac{0}{\|t_i\|}|$$

Les indices étant égaux à 0, le dernier tableau de pixels comparé sera retenu :

$$\min(c_0, c_1, ..., c_{l_{buff}-1}, c_{l_{buff}})$$

A chaque fois, ve, le tableau de pixels de référence sera le tableau dont le premier pixel sera le dernier de la zone délimitée. Son premier pixel sera à la coordonnée (x_{ve}, y_{ve}) où, dans le cas horizontal, $y_{ve} = y_{ms}$, et $x_{ve} = x_{ms} + l_{ms}$, et, dans le cas vertical, $x_{ve} = x_{ms}$, et $y_{ve} = y_{ms} + h_{ms}$.

L'opération de sélection ne sert donc à rien. Nous avons proposé, dans la section 3.3.2, une solution utilisant la variance des vecteurs qui n'a pas été testée. Cette solution devrait apporter une amélioration dans la sélection du vecteur le plus contrasté. Mais nous n'avons pas pu chiffrer le gain concret.

Le plus gros inconvénient reste le temps de traitement. Il est en effet d'autant plus important que les images sources sont grandes et que les paramètres sont imprécis. Plusieurs solutions peuvent être mises en oeuvre pour améliorer ce temps.

La première proposition repose simplement sur l'introduction d'un pool de thread au niveau du calcul des critères de comparaison entre les tableaux de pixels. Il est en effet possible d'implanter au sein du calcul un ensemble de threads qui effectue le calcul et qui rapporte le résultat. Il reste à déterminer quelle politique mettre en place :

- un pool de threads de taille fixe, chaque thread puise dans une liste de tâches, les arguments du calcul à effectuer ;
- un pool de threads de longueur variable, en fonction du nombre de tâches à réaliser.

Cette première proposition ne nécessite pas énormément de modifications au sein de l'architecture de l'application.

Une seconde proposition serait de développer une architecture de "grid-computing". L'idée se positionne dans la lignée de la solution basée sur des threads, à la différence que chaque thread serait remplacé par un processus sur des machines différentes. La clinique offre en effet une multitude de ressources, lesquelles ne sont pas constamment entièrement utilisées.

Pour chacune de ces propositions, il est très facilement possible de créer un module client-serveur qui reprendrait les méthodes particulièrement gourmandes en temps de calcul. Plusieurs découpes sont envisageables :

- une découpe qui enverrait deux tableaux de pixels, le module worker calculerait le résultat de la comparaison entre les deux tableaux.
- une découpe qui enverrait un tableau de pixels de l'image maître et un ensemble de tableaux de pixels de l'image esclave. Le worker calculerait les comparaisons entre le vecteur étalon de l'image maître et chacun des vecteurs de l'image esclave repris dans l'ensemble.

Un autre problème concernant la création de méga-images, est lié à la plate-forme Analysis qui supporte la méthode de création. En effet, suite au changement de caméra, la platine a dû être reconfigurée. Pour cela, comme C. Peeters a déjà pu le faire apparaître dans ses annexes, Analysis propose d'effectuer une calibration automatique lors du placement de la platine. La caméra semblait être mal positionnée. Aucune coregistration ne réussissait. Nous n'avons pas pu résoudre ce problème, les images étaient générées dans un sens différent. Il serait intéressant de clarifier cette situation et d'expliciter toutes les démarches à effectuer pour obtenir un système opérationnel clair et précis.

4.4 Extraction et envoi des résultats d'analyses

L'extraction des résultats d'analyses stockés dans les .mdb générés par CellaVision est à présent en place. Les développements effectués durant le stage et la préparation de ce mémoire ont permis de fournir une application fonctionnelle, utile et surtout utilisable par les laborantins. En effet, "l'extracteur" est utilisé tous les jours par les membres de l'équipe du laboratoire.

L'avantage de cette application est qu'elle permet d'archiver de manière centralisée les images issues des analyses effectuées. Cette centralisation permet aux experts qui sont en dehors du laboratoire de consulter et visualiser les résultats via l'application "Telemis". Pour l'heure, cette fonctionnalité est très utile dans le contexte des gardes. En effet, certains membres du laboratoire ont des gardes à effectuer c'est-à-dire qu'ils doivent être disponibles pour les urgences. Auparavant, ces personnes devaient se rendre au laboratoire pour donner un avis sur les analyses effectuées. Grâce à cette application, il est possible de visualiser les résultats de chez soi via "Telemis"(voir figure 4.1), ce qui permet de gagner du temps et du confort dans les situations d'urgence.

Ajoutons que, l'application orientée visualisation a un usage bien plus ciblé. En effet,

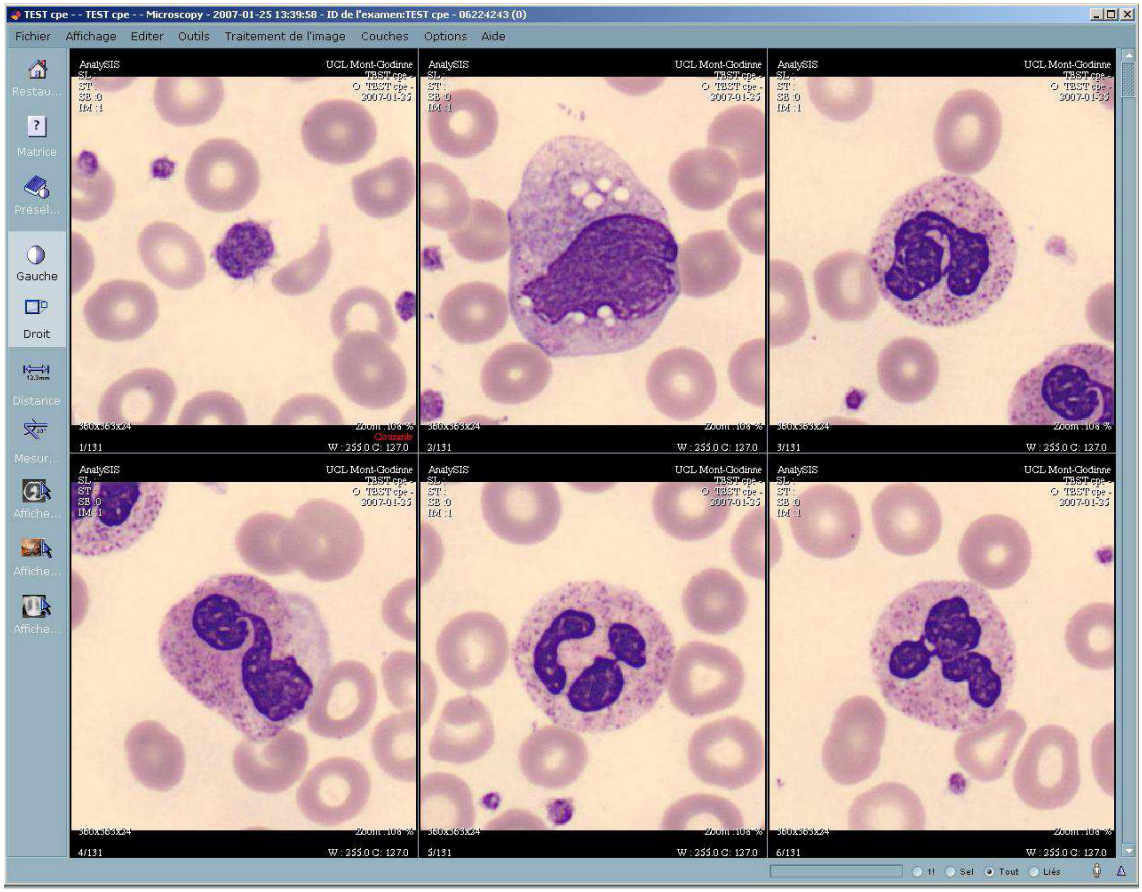

FIG. 4.1 – Interface de "Telemis"[Pee07].

l'avantage de l'application orientée service est qu'elle participe à une politique de partage d'informations et de centralisation autour du dossier de patient. Mais la partie visualisation est très intéressante dans l'optique d'une vérification de contenu de base de données. En effet, nous l'avons vu, une fois les bases de données d'archives générées, celles-ci n'étaient plus lisibles à cause, d'une part, de la complexité des tables du fichier MS Access et, d'autre part, de la complexité des données archivées (images en données "binaires", chaîne de caractère,...). Cette complexité est encore moins évidente à comprendre pour des utilisateurs non-initiés. Ainsi, l'application de visualisation permet, à présent de consulter le contenu de la base de données. Même si elle n'est pas utilisée tous les jours, il est rassurant de la savoir opérationnelle. De plus, une fois qu'il a visualisé le contenu, l'utilisateur peut, à présent, transmettre les images qu'il a sélectionnées à un serveur DICOM distant. Ce qui permet de diminuer les données archivées et de ne conserver que les images pertinentes.

Cependant, bien que fonctionnels, ces services restent très rudimentaires. Ainsi, pour le moment, le service et l'interface ne sont pas très robustes. Plus particulièrement pour

le service[1], si le service est en cours d'envoi, il ne faut en aucun cas l'interrompre. Une interruption suivie d'un redémarrage provoquerait une redondance des images envoyées sur le serveur DICOM. Le service envoie les "DICOM objects" sur le serveur sans vérifier si l'image s'y trouve déjà.

Trois solutions peuvent être envisagées pour y remédier :

- la première consiste à utiliser une possibilité du réseau DICOM déployé à Mont-Godinne, le "storage-commitment" ;
- la deuxième proposition consiste à mettre en place un fichier persistant lié au service d'extraction. Ainsi, ce fichier contiendrait les informations sur les images à envoyer ;
- la troisième solution est d'inscrire les objets DICOM dans un dossier d'envoi tant qu'ils n'ont pas été réceptionnés.

La première solution consiste à mettre en place un mécanisme qui va interroger le serveur DICOM distant afin de savoir si l'objet DICOM à envoyer a déjà été reçu. Si la réponse est négative, il transmettra les données, dans le cas contraire, il ne fera rien et traitera la donnée suivante. Cette solution nécessitera des modifications dans l'organisation du service d'envoi. Cette communication entre le serveur DICOM est asynchrone. Il faudra donc mettre en place un "serveur" du coté service d'extraction auquel on adressera les réponses du serveur DICOM.

La deuxième solution semble plus simple à mettre en place. L'idée est de conserver un fichier d'état d'envoi des images. Pour cela, lors de l'étape d'extraction, les images sont enregistrées temporairement sur le disque. Pour chaque image écrite temporairement, on ajoute une ligne dans un fichier contenant les informations de l'image et sa référence sur le disque. Le thread d'envoi est alors responsable de créer un objet DICOM sur base d'une ligne de ce fichier et d'envoyer cet objet sur le serveur. Lorsque le stockage de celui-ci a été confirmé, il reste à supprimer la ligne le référençant. L'inconvénient de cette solution est que les images temporaires peuvent être supprimées sans qu'elles aient été envoyées. Elles sont alors toujours référencées par le fichier d'état d'envoi. Lors de l'envoi, l'insertion de l'image dans l'objet DICOM ne pourra réussir en raison de l'absence de l'image.

La troisième solution est proche de la précédente. Elle part du problème lié à la suppression des images temporaires. Une solution pourrait être de remplacer le fichier d'état d'envoi par un dossier contenant des objets DICOM à envoyer. Il est en effet possible d'écrire des objets DICOM temporairement sur le disque. C'est donc ici le service d'extraction qui est responsable de la création des objets DICOM. Le thread d'envoi est, lui, chargé de parcourir le dossier d'objet DICOM, de les envoyer un à un sur le serveur DICOM et de les supprimer si l'envoi a réussi.

Ces deux dernières solutions n'empêchent évidemment pas d'extraire et d'envoyer avec succès le contenu d'une base de données MS Access et d'effectuer la même démarche une seconde fois. La première solution peut faire face à ce cas particulier mais la complexité de mise en œuvre est plus importante.

[1] L'interface n'étant utilisée que pour permettre de visualiser le contenu des bases de données d'archives, elle est moins sensible au problème concernant l'envoi et la redondance sur le serveur.

En parallèle à toutes ces considérations techniques et relatives à la performance du système d'archives, il serait intéressant d'améliorer l'interface utilisateur. En effet, l'interface du service n'est qu'une icône disponible dans le "System Tray" (en français, zone de notification). Elle a été développée dans l'esprit du service qui fonctionne en batch, en parallèle avec d'autres applications. Mais le principe du service est-il la bonne solution ?

De plus, il serait intéressant de se séparer de l'utilisation de .bat. Cette technique utilise la console MS-Dos ce qui est perturbant pour l'utilisateur. En effet, un processus qui fonctionne constamment ne devrait pas être visible constamment. D'où l'idée d'utiliser la zone de notification dans la barre des tâches pour fournir à l'utilisateur une icône l'informant sur l'état de fonctionnement de l'application.

Pour le moment, les images sont stockées sur un serveur. L'étape suivante consiste à créer le lien avec le dossier informatique du patient de manière à rendre disponible le contenu des analyses à l'ensemble du corps médical ayant accès à ce dossier.

A côté de l'envoi du contenu de base de données, les responsables du laboratoire d'hématologie de Mont-Godinne ont aussi demandé s'il était réalisable d'envoyer des images solitaires sur un serveur. Leur idée était de rendre consultable à distance des images telles que des numérisations d'échantillons de mœlle ou autre. Cette application peut être très facilement mise en place. Il suffirait d'une interface offrant un explorateur de façon à sélectionner l'(ou les) image(s) à envoyer et d'en créer les objets DICOM à envoyer. Le thread qui envoie les objets une fois créés, peut être réutilisé.

L'interface devrait aussi fournir des champs à initialiser de manière à ce que l'image dispose d'informations permettant de l'identifier sur le serveur.

Il est cependant nécessaire que le laboratoire puisse s'entendre avec le service de médecine nucléaire et le service informatique concernant l'usage qui sera fait du serveur mis à disposition.

4.5 Caractérisation de globules blancs

Le travail de caractérisation de globules blancs n'est, bien entendu, pas arrivé à son but. Le temps a manqué pour permettre la mise en place d'un système fonctionnel. Nous avons seulement pu débuter une synthèse des différents globules blancs et la recherche de méthodes à implémenter pour permettre à un ordinateur de les identifier.

La caractérisation de globules blancs a été entamée par les travaux de C. Peeters. La méthode imaginée était basée sur l'extraction de caractéristiques par le traitement d'images de globules blancs repérés. Il est donc nécessaire pour chaque classe de globules de définir des règles dérivées des caractéristiques extraites. Une fois ces règles définies, aucune autre information n'est nécessaire.

Nous avons vu qu'il était possible d'extraire des caractéristiques par le traitement des méga-images. Ainsi, l'architecture de l'application de caractérisation a été modifiée et une tendance a été tracée. Nous avons mis en place une première classification sur base des critères déjà extraits (pour rappel, l'application différencie les plaquettes et les globules blancs). Autre point, nous avons équipé l'interface d'une option permettant à l'utilisateur de préciser l'objectif utilisé pour la capture des images sources dans le but de permettre

l'utilisation de l'application sur des images sources acquises sous différents objectifs (40x, 60x, 100x,...).

Plus largement, l'organisation de l'architecture a été modifiée pour permettre l'utilisation de l'application dans le traitement d'autres types d'images. Ainsi, il pourrait être facilement possible de développer des règles pour caractériser les composants d'un échantillon de moelle et d'ajouter ce "module" à l'application actuelle. Des changements devront encore s'ajouter en vue de poursuivre dans cette voie.

Nous l'avons vu, les avancées réalisées sur l'extraction proprement dite n'ont pas été nombreuses. Les méthodes d'extraction ont donné quelques résultats, et la méthode de Canny pourrait donner satisfaction pour certains types de globules blancs. Cependant, il vaudrait peut-être mieux une méthode qui soit indépendante du type de globules blancs. Les résultats sont donc à approfondir.

Ajoutons que des recherches effectuées ont fini par révéler des travaux portant sur le même sujet. Ces documents ont permis de recouper et confirmer les observations déjà effectuées concernant les critères ayant un intérêt pour une classification de globules blancs. Ces travaux sont présentés dans la thèse de doctorat de J. Angulo intitulé "Morphologie mathématique et indexation d'images couleur. Application à la microscopie en bio-médecine" ([Ang03]).

Il confirme certains critères choisis dans ce mémoire pour classifier les globules blancs. Il propose aussi des méthodes de traitements des images pour l'extraction des différents contours et la caractérisation de la couleur (plus particulièrement l'intensité lumineuse).

Une autre solution peut être envisagée. Celle-ci est imaginée à partir des travaux de Raphaël Marée. Ces travaux ont porté sur la classification d'images de manière générale. Sans aller dans les détails, la méthode développée et le logiciel mis en place permet de lier une image à un type. La création de ce lien est basé sur une base de données d'images. La première étape consiste à extraire de ces images des "sous-images".

Les contacts avec Raphaël Marée m'ont permis de me procurer le logiciel Pixit (figure 4.2) de l'entreprise Pepite S.A. située à Angleur. Ce logiciel implémente la méthode décrite ci-dessus. Il permet ainsi de définir un ensemble d'images servant de base de connaissances (cet ensemble est appelé "learning set", et serait dans notre cas, des images de globules blancs dont la classification a été déclarée), de lui soumettre de nouvelles "connaissances" (dans notre cas, des images de globules blancs) et d'obtenir une classification par rapport à la base de connaissances. Nous avons pu effectuer quelques rapides tests sur base de bases de données de 1890 images issues de CellaVision. Les résultats sont assez positifs.

Il serait intéressant de présenter la base de connaissance utilisée à des experts pour s'assurer des classifications déterminées par CellaVision. En effet, l'ensemble d'images classifiées dont nous disposions contenaient 312 images de classe "indéfinie". De plus, nous avons utilisés les paramètres par défaut, un ajustement de ceux-ci offrirait peut-être un meilleur résultat.

Ces premiers résultats ont encore un taux d'erreur relativement haut. Cependant, l'équipe de Raphaël Marée ont réalisés dans le passé des tests sur des images de type biologique qui semblent laissé penser que de meilleurs résultats pourraient être obtenus.

FIG. 4.2 – Capture d'écran de résultats avec Pixit

Ainsi, la figure 4.3 montre un échantillon d'un jeu de tests utilisés sur des globules rouges. La base de connaissance était composée de 5062 images de globules rouges, séparées en 3 classes par un expert : 916 echinocytes, 3259 stomatocytes et 887 discocytes. Plusieurs tests avec des cellules sanguines (plus précisément, des globules rouges) ont déjà été effectués par l'équipe de développement (cfr [MGW06]) et ont apportés des résultats où 80% des cellules étaient bien classifiées. Ce pourcentage est au moins égal au taux de réussite pour un humain.

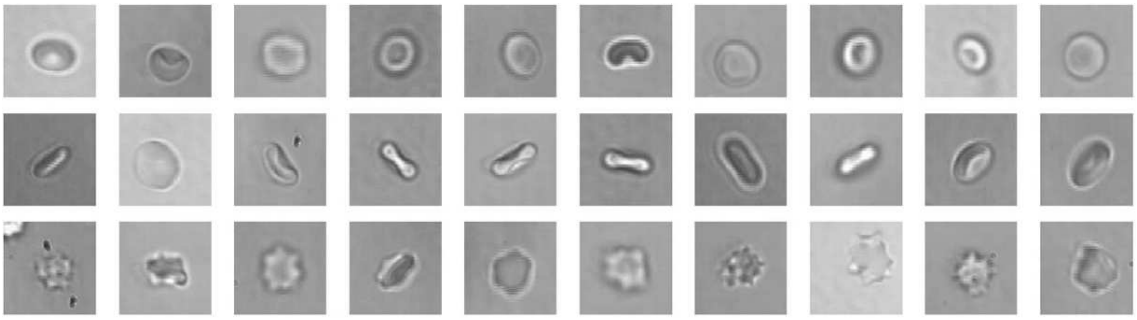

FIG. 4.3 – Globules rouges : stomatocyte (1^{ere} ligne), discocyte (2^{eme} ligne) et echinocyte (3^{eme} ligne) (Image de [MDGW07])

Chapitre 5

Conclusion

La télémédecine est une notion à laquelle de nombreuses technologies pourront encore servir. Au fil des années, elle s'étoffera de gadgets technologiques participant à la mise en place d'un système beaucoup plus large.

Depuis de nombreuses années, des étudiants en informatique participent à la recherche sur la mise en place d'un système automatique pour supporter l'analyse cyto-hématologique du laboratoire. Ainsi, l'automatisation a été introduite à plusieurs étapes du processus. Le premier niveau touché est la capture de petites images à partir de lames de frottis sanguins.

Ensuite, une fois ces images numérisées, un système de coregistration a été développé permettant ainsi de travailler sur des grands champs. Ces méga-images ont subi des améliorations suite à des recherches portant sur le format de compression utilisé pour leur stockage et la qualité intrinsèque. Les recherches ont aussi porté sur la localisation des globules blancs dans ces méga-images et, de manière plus discrète, sur leur classification en usant de différentes techniques telles que filtrage, binarisation,... Au fil des années, l'ensemble de ce système de télémicroscopie s'est organisée en une architecture client-serveur et suit une politique dite de "store-and-forward".

Nous avons débuté en présentant l'environnement de travail ainsi que les développements effectués précédemment. Ces derniers faisaient appel à diverses technologies telles que la communication RS232 pour la communication entre les différents composants du système, le codage JPEG2000 pour la compression des images de frottis sanguins, une infrastructure client-serveur pour la gestion et l'organisation du système et de multiples techniques de traitement d'images numériques pour la localisation et l'extraction des globules blancs.

Par la suite, les multiples objectifs de mon stage ont été élicités. Nous avons fourni les outils pour comprendre les problèmes, les méthodes et les technologies utilisées. Nous avons présenté les différents résultats pour chacun des objectifs. La dernière partie propose un bilan des travaux et des résultats obtenus ainsi qu'une liste d'idées pouvant mener à l'amélioration du système actuel.

Le premier objectif portait sur un problème d'autofocus et n'a pas trouvé de véritable solution. Mais le travail fourni a permis de comprendre le fonctionnement de la plate-forme

d'acquisition et plus précisément du microscope et de ses composants. On peut, à présent, écarter certaines pistes envisagées et se consacrer à de nouvelles propositions de solutions.

Ce mémoire a aussi présenté la modification de la méthode de coregistration développée précédemment afin de fournir au laboratoire un système plus souple et moins dépendant de la résolution des images sources.

Nous avons aussi poursuivi la recherche dans le cadre de la conception d'une application de caractérisation et classification de globules blancs. Les travaux ont permis d'effectuer un "recensement" des classes de globules blancs et d'avancer certaines pistes pour l'extraction des caractéristiques de ces classes. Ils ont aussi permis de fournir un début de classification à partir des critères déjà extraits.

A côté de ce système de classification, le laboratoire d'hématologie avait émis le souhait de pouvoir visualiser et surtout archiver les contenus de bases de données d'images d'analyse. Ce souhait avait donné naissance à une petite application lors de travaux précédents mais elle n'avait pu être terminée. Elle est, à présent, fonctionnelle et utilisée quotidiennement par les laborantins, leur offrant un archivage centralisé et, grâce à l'utilisation de "Telemis" et du standard DICOM, une source de partage des résultats d'analyses.

Le stage qui m'a été proposé et le mémoire qui en découle, ont été très enrichissants pour moi. Cela m'a permis de découvrir le monde des soins de santé en général, mais plus précisément l'hématologie et la position centrale qu'elle occupe dans l'émission d'un diagnostic.

Le travail de synthèse et de rédaction de ce mémoire a été aussi l'occasion de réaliser que la littérature est parfois bien loin de la technique. Il est bien difficile d'exprimer des notions comprises et très techniques et d'en rendre la compréhension abordable aux lecteurs potentiels.

Annexes et bibliographie

Annexes

6.1 Résultats des coregistrations

Cette annexe a pour but d'exposer les résultats obtenus par la coregistration. Elle est séparée en deux parties, la première illustrent les résultats obtenus avec la méthode précédente, la seconde, présentent les résultats de la nouvelle méthode.

6.1.1 Coregistration avec l'ancienne méthode (cfr 2.3)

La première partie pour illustrer la coregistration obtenue avec l'ancienne méthode de coregistration.

Deux résultats sont proposés :

1. Le premier résultat est obtenu sur base des images sources capturées avec l'ancienne caméra, la caméra Sony PowerHAD.
2. Le second résultat est obtenu sur base des images sources capturées avec la nouvelle caméra, la caméra Olympus DP71.

Résultat obtenu avec les images sources de la caméra Sony PowerHAD

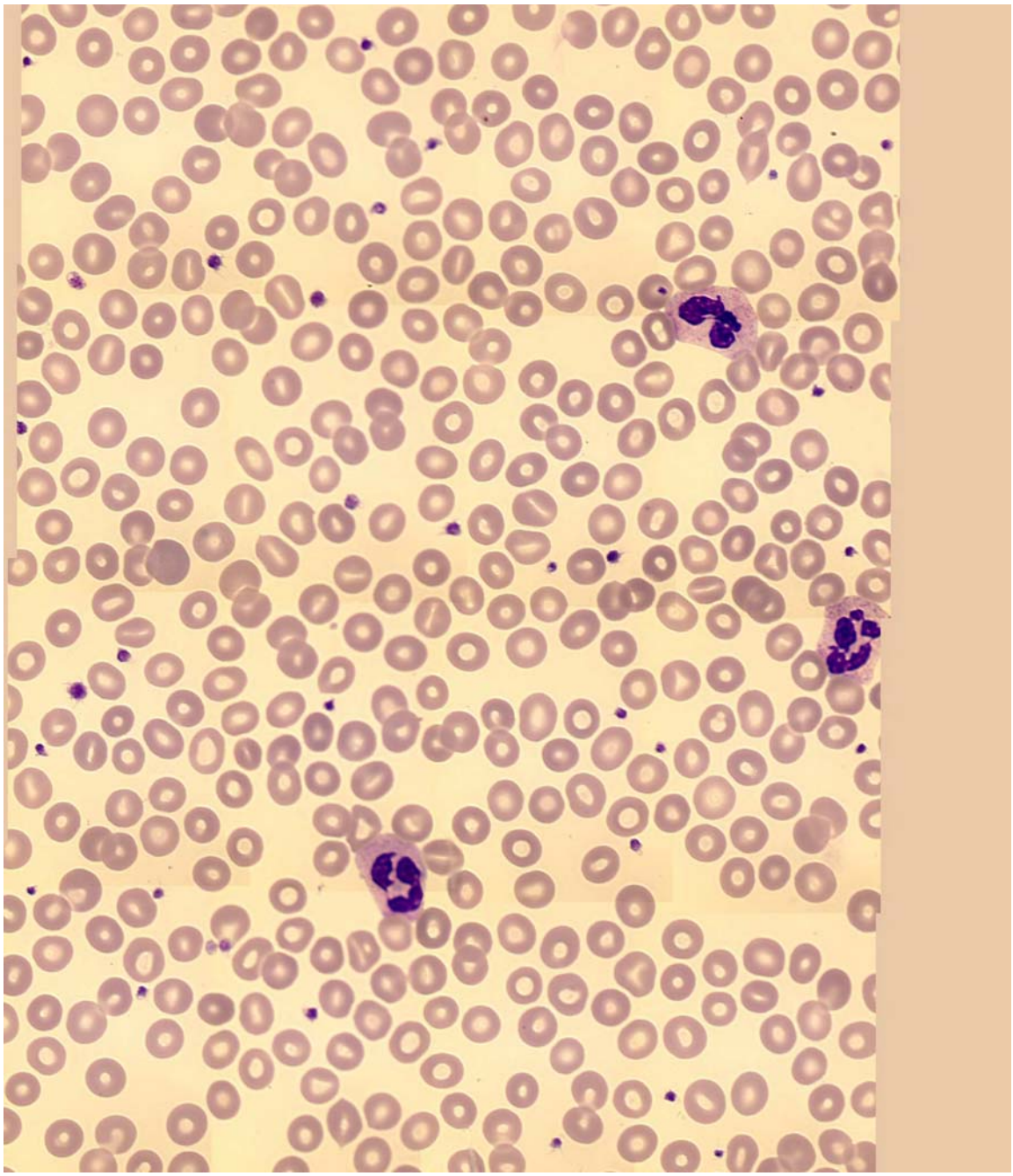

FIG. 6.1 – Coregistration des images sources issues de Sony PowerHAD avec l'ancienne méthode

Résultat obtenu avec les images sources de la caméra Olympus DP71

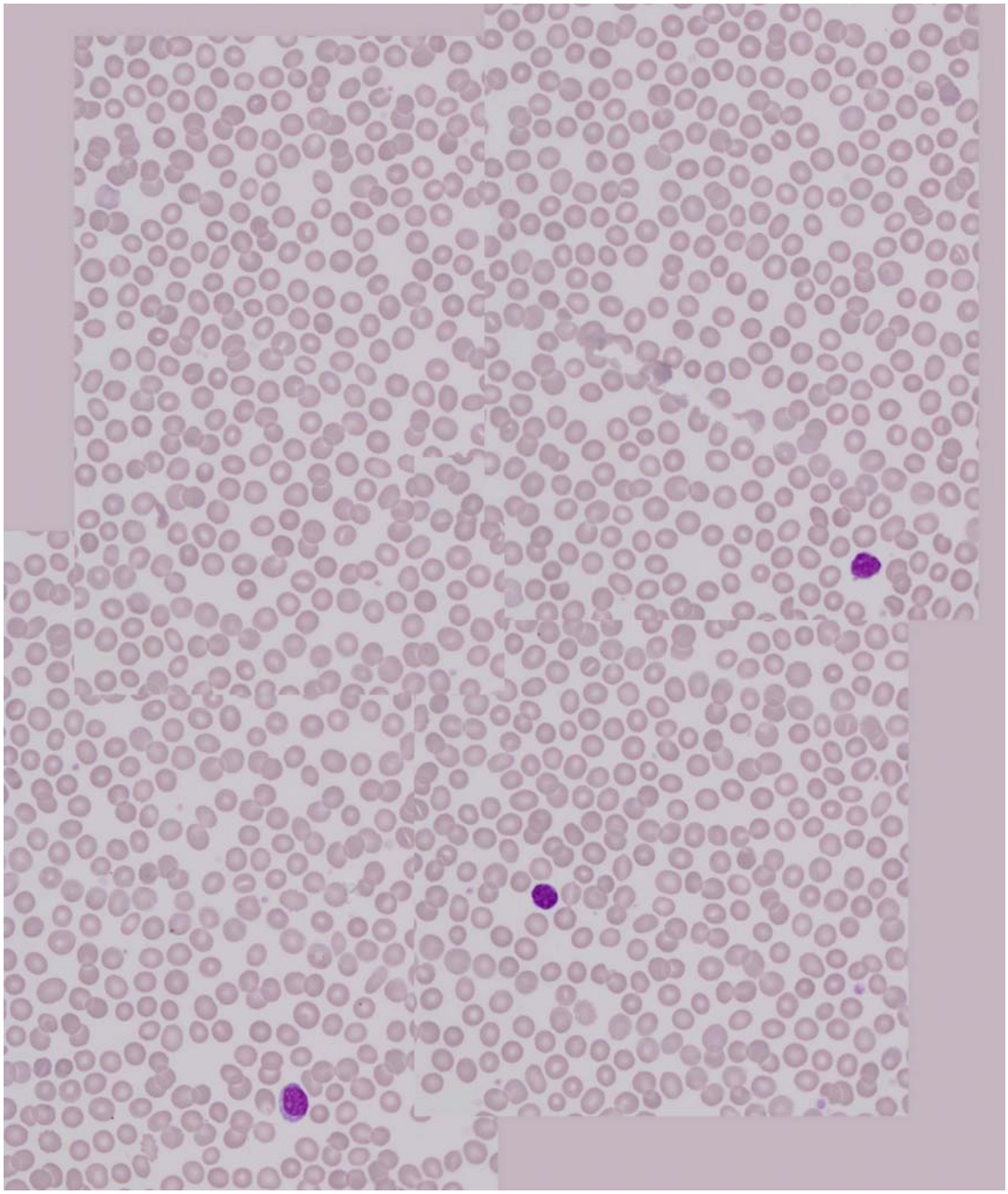

FIG. 6.2 – Coregistration des images sources issues de Olympus DP71 avec l'ancienne méthode

6.1.2 Coregistration avec la nouvelle méthode (cfr 3.3)

Trois résultats sont proposés :

1. Le premier résultat est obtenu sur base des images sources capturées avec l'ancienne caméra, la caméra Sony PowerHAD.
2. Le second résultat est obtenu sur base des images sources capturées avec la nouvelle caméra, la caméra Olympus DP71.
3. Le troisième résultat sort du champ de l'hématologie. Les images ont été constituées manuellement par captures d'écran successives de cartes issues de Google Maps [1]. Il s'agit de captures satellite de la région de Rome.

[1] `www.http://www.google.fr/maps`

Résultat obtenu avec les images sources de la caméra Sony PowerHAD

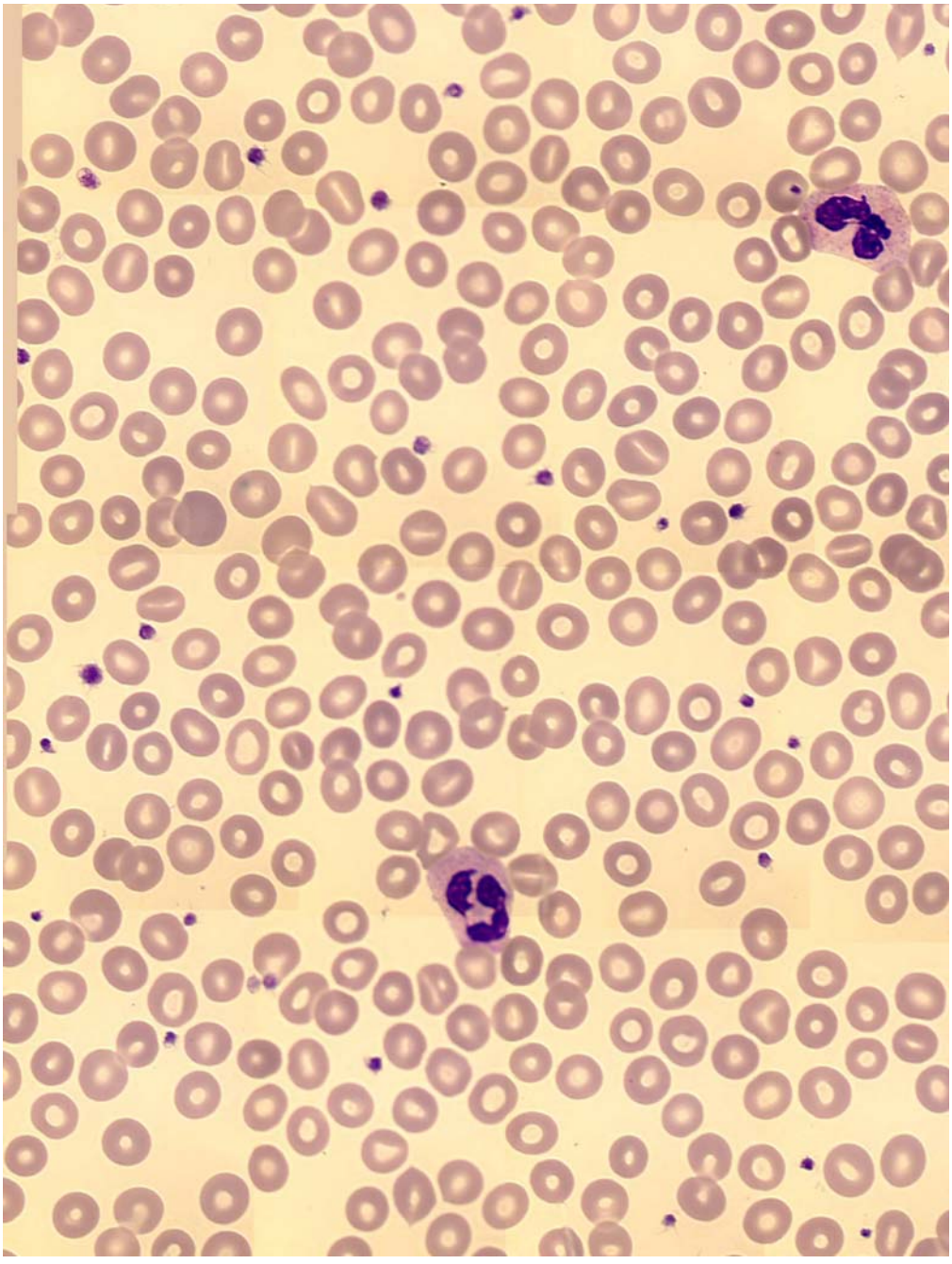

FIG. 6.3 – Coregistration d'images sources de Sony PowerHAD (nouvelle méthode)

Résultat obtenu avec les images sources de la caméra Olympus DP71

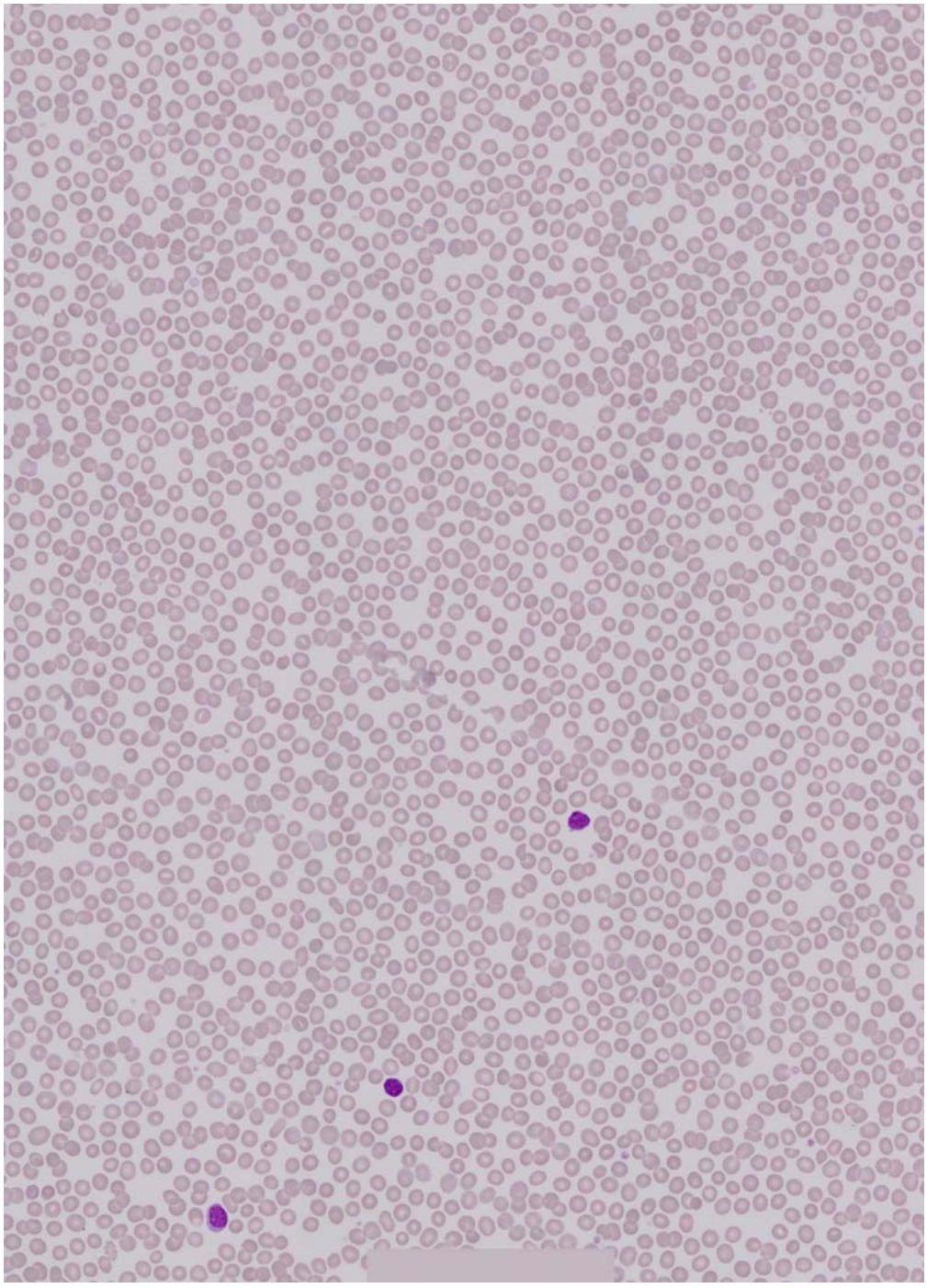

FIG. 6.4 – Coregistration d'images sources de Olympus DP71 (nouvelle méthode)

Résultat obtenu avec les images sources de Google Maps

FIG. 6.5 – Coregistration de capture d'écran de Google Maps (nouvelle méthode)

6.2 Mise en place et démarrage du coregistreur

Afin d'exécuter le coregistreur (c'est-à-dire l'ensemble composé du serveur de coregistration et d'un client de coregistration), il faut vérifier les paramètres contenus dans le fichier nommé *serverCreation.ini*. Celui-ci doit se trouver dans le dossier *"./newCreation/bin/"*.

Les paramètres du serveur sont séparés en deux parties :

1. Une partie pour les propriétés du serveur qui doit comprendre :
   - la section *[cache]* sert à identifier les propriétés du cache du serveur
     * l'entrée TILEPOOL_CAPACITY permet de définir le nombre de tiles pouvant être contenues dans le cache (par défaut, 10) ;
     * l'entrée SOURCECACHE_CAPACITY permet de définir le nombre d'images source (tiff de +/- 1Mo) que le serveur peut garder en mémoire (par défaut, 15).
   - la section [Encoder] permet de définir les options de l'encodeur de jpeg2000
     * l'entrée JJ2000_OPTION contient les paramètres utilisés par le thread encodeur JPEG2000(par défaut, "-lossless off -rate 1 -tiles 256 256")
   - la section [INPUT] permet de spécifier le chemin du dossier contenant les images sources
     * l'entrée FILE_ROOT contient le chemin menant aux images. Attention, les images sources sont dans une hiérarchie particulière qui s'articule comme tel : <chemin standard>/<userID>/<imageName>. Le chemin à spécifier est donc <chemin standard>.
2. Une autre partie pour les propriétés des images à coregistrer :
   - la section [<RegistrationName>] pour définir les propriétés des images dont la registration est "Registration" (ce nom devra être envoyé par le client pour que le serveur sache quelles propriétés utiliser). Le fichier de paramètres contient autant de sections de ce type qu'il a de formats d'images sources à coregistrer, chaque format étant unique et identifié par la nom de la section <RegistrationName>.
     * l'entrée LAG_MAX_HORI permet de définir un décalage vertical maximum entre deux images positionnées horizontalement côte à côte[2] ;
     * l'entrée LAG_MAX_VERT permet de définir un décalage horizontal maximum entre deux images positionnées verticalement côte à côte[3] ;
     * l'entrée OVERLAP_MAX_HORI permet de préciser l'overlap maximum entre deux images l'une à droite de l'autre[4].
     * l'entrée OVERLAP_MAX_VERT permet de préciser l'overlap maximum entre deux images l'une en dessous de l'autre[4].
     * l'entrée FACTOR_SEEK_ORIGINAL permet de préciser le facteur qui divisera la largeur ou la hauteur de l'image pour obtenir un recouvrement par défaut (respectivement *OVERLAP_MAX_HORI* ou *OVERLAP_MAX_VERT*) lorsque ce dernier est inconnu.

[2] Sur l'axe vertical (horizontal), l'image du bas sera décalée vers la droite ou vers la gauche, et donc un décalage horizontal. Notons que cette valeur ne peut pas être supérieure à la longueur de l'image source.

[3] Sur l'axe horizontal, l'image la plus à droite peut être décalée vers le bas ou vers le haut, et donc un décalage vertical. Notons que cette valeur ne peut pas être supérieure à la hauteur de l'image source.

[4] Si l'overlap est inconnu, l'utilisateur peut entrer la valeur " -1 ", valeur par défaut

Une fois le fichier de configuration correctement rempli, le serveur peut être démarré via un fichier ".bat" nommé "serverCoregistration.bat". Les paramètres de ce fichier doivent aussi être vérifiés et sont :

1. le numéro de port ouvert et utilisé par le serveur
2. le nombre de sessions pouvant être traitées simultanément par le serveur

Ajoutons qu'il est important que le paramètre java "-mx900000000" soit présent dans la commande Java du fichier ".bat" afin d'obtenir les ressources mémoire nécessaires à la création des méga-images.

Lorsque le serveur a été démarré, il reste à lancer le client de coregistration. Celui-ci peut être démarré via un fichier ".bat", client.bat. Ce fichier doit être configuré sur base des images sources à coregistrer. Ces paramètres sont, dans l'ordre, :

1. l'adresse ip du serveur de coregistration ;
2. le port ouvert sur le serveur de coregistration
3. le nom de la section des paramètres de coregistration (voir les paramètres du serveur de coregistration, [<Registration>]) ;
4. le nom de l'utilisateur (voir l'entrée "FILE_ROOT" dans les paramètres du serveur de coregistration, <userID>) ;
5. le nom de la mega-image (voir l'entrée "FILE_ROOT" dans les paramètres du serveur de coregistration, <userID>) ;
6. la longueur de la méga-image exprimée en nombre d'images sources ;
7. la longueur de la méga-image exprimée en nombre d'images sources ;
8. le coefficient d'homogénéisation que le serveur doit appliquer à la méga-image.

6.3 Mise en place et démarrage du convertisseur

Le convertisseur permet de transformer les images sources dont le format de fichier et de nom ne correspond pas au format pris en charge par le serveur de coregistration. Ces images sources doivent être identifiées par un préfixe composé d'un nom d'image commun à chacune d'elles et d'un suffixe unique numérique définissant leur ordre de capture.

Exemple : si l'ensemble des images sources est composé de " image1.jpg ", " image2.jpg ", " image3.jpg ", " image4.jpg " et qu'elles ont été générées de gauche à droite suivi de droite à gauche et de haut en bas, le convertisseur les transformera pour donner " image[0-0].tif ", " image[1-0].tif ", " image[0-1].tif ", " image[1-1].tif "

Les arguments à passer au programme sont : *java -jar converter.jar <constante du sens horizontal> <constante du sens vertical> <nom commun à toutes les images> <format des images à créer> <chemin d'accès à la première image à transformer> <nombre de lignes> <nombre de colonnes>*

Les différentes constantes à utiliser ont les valeurs suivantes :

- Pour le sens horizontal :
  - Si les images ont été générées de droite à gauche pour chaque ligne, la constante est 0 ;
  - Si les images ont été générées de gauche à droite pour chaque ligne, la constante est 1 ;
  - Si les images ont été générées de droite à gauche pour la première ligne et inversement pour la seconde et ainsi de suite, la constante est 2 ;
  - Si les images ont été générées de gauche à droite pour la première ligne et inversement pour la seconde et ainsi de suite, la constante est 3 ;
- Pour le sens vertical :
  - Si les images ont été générées de haut en bas, la constante est 4 ;
  - Si les images ont été générées de bas en haut, la constante est 5 ;

Les autres arguments sont :

- <nom commun à toutes les images>, le nom commun à toutes les images à reformater, dans l'exemple ci-dessus, cet argument serait "image" ;
- <format des images à créer>, peut être "tif", "jpeg", "png" ou "pnm" ;
- <chemin d'accès à la première image à transformer>, sous Windows, par exemple, "c :/image/Image1.jpg" ;
- <nombre de lignes> sera, dans l'exemple ci-dessus, 2 (nombre de lignes dans la matrice de la méga-image) ;
- <nombre de colonnes> sera, dans l'exemple ci-dessus, 2 (nombre de colonnes dans la matrice de la méga-image) ;

Le fichier .bat, pour l'exemple ci-dessus, sera de la forme :

java -jar converter.jar 3 4 image tiff c :/Image/image1.jpg 2 2

6.4 Mise en place et démarrage de l'extracteur

Une première vérification doit être effectuée pour le démarrage de l'extracteur tant en mode service qu'en mode visualisation. Il faut que certains fichiers soient présents dans les dossiers de l'application.

– Il est nécessaire que le fichier[5] "parameter.ini" se trouve dans le dossier "extractor-FromBD binsous la forme

```
[extractor]
delayBetweenTransmission=10000
repertoireInput=
addressDicom=adresse par defaut
deletePolicy=8
outputDirectoryTemp= <dossier cible>
```

FIG. 6.6 – Paramètres de l'extracteur

  ◦ `delayBetweenTransmission` contient la valeur en ms (millisecondes)entre chaque " check " dans le dossier ;
  ◦ `repertoireInput` contient l'adresse du répertoire dans lequel on va aller chercher les bd à extraire ;
  ◦ `adressDicom` contient le nom du serveur DICOM dernièrement utilisé ;
  ◦ `deletePolicy` contient une constante de politique de suppression des données (BD et images extraites). Il peut prendre les valeurs suivantes :
    – 5 si on veut supprimer les images et les BD ;
    – 6 si on veut conserver les images (et supprimer les BD) ;
    – 7 si on veut conserver les BD (et supprimer les images) ;
    – 8 si on veut conserver les BD et les images.
  ◦ `outputDirectoryTemp` contient l'adresse du répertoire duquel seront extraites les images (temporaires).

– Il faut aussi s'assurer de la présence du fichier " aet.cfg " dans le dossier " extractor-FromBD bin ". Il est indispensable pour l'envoi en DICOM.
– Il faut aussi le dossier "images" dans le " extractorFromBD ".

Pour lancer l'extracteur avec l'interface de visualisation du contenu de la BD :

– Double cliquez sur "Run ExtractorIHM.bat" ;
– Une fois l'application démarrée, choisissez le fichier à extraire.

FIG. 6.7 – Chemin de la base de donnée en entrée de l'extracteur

– Sélectionnez ensuite le dossier destination temporaire
– Appuyez sur "Extract images"

[5]Ce fichier n'est pas particulièrement important pour l'extractor avec IHM, il l'est beaucoup plus pour l'extractor en Service.

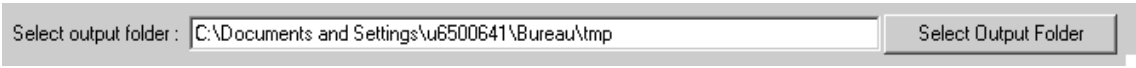

FIG. 6.8 – Chemin du dossier temporaire des données extraites

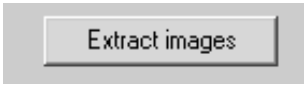

FIG. 6.9 – Bouton de lancement de l'extraction

– Les images sous extraites et visibles dans la galerie.

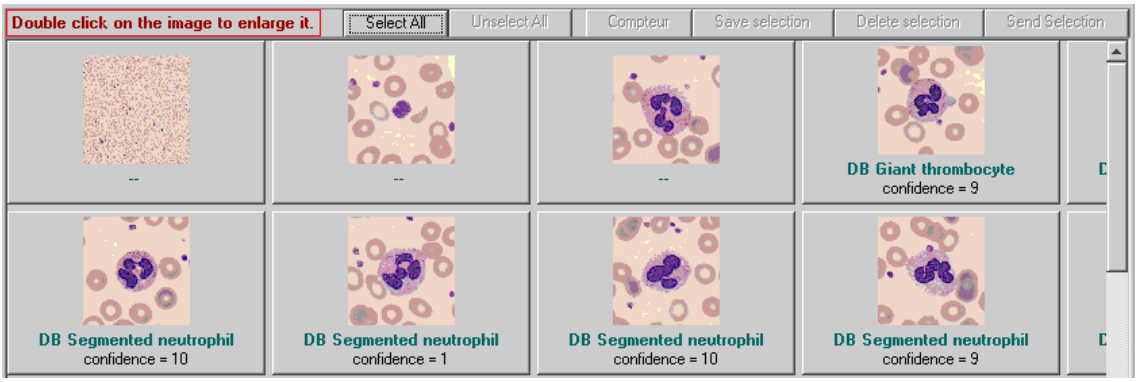

FIG. 6.10 – Galerie des images extraites

– Pour les envoyer en DICOM sélectionnez les images à envoyer, et cliquez sur "Send Sélection". Il apparaît alors une fenêtre de configuration de serveur destination Ainsi

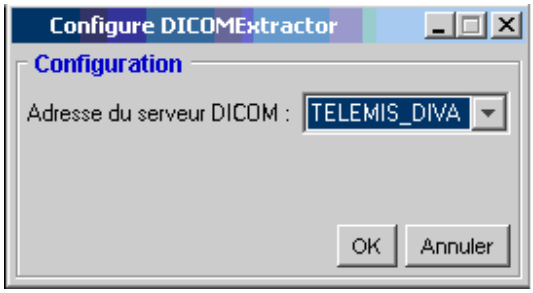

FIG. 6.11 – Sélection du serveur DICOM

qu'une fenêtre[6] propre à l'envoi DICOM (permet de voir les logs des connexions au serveur DICOM),
– Sélectionnez "OK" dans la fenêtre de configuration pour valider l'envoi vers le serveur.

[6]Attention, la fermeture de cette fenêtre entraîne la fin du programme, ne pas la fermer si vous ne désirez pas quitter le programme.

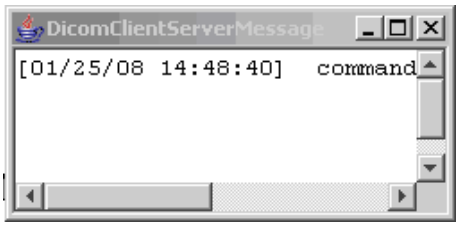

FIG. 6.12 – Fenêtre spécifique à DICOM

– En arrière plan, il est possible de voir l'évolution de l'envoi des objets Dicom dans la console DOS.

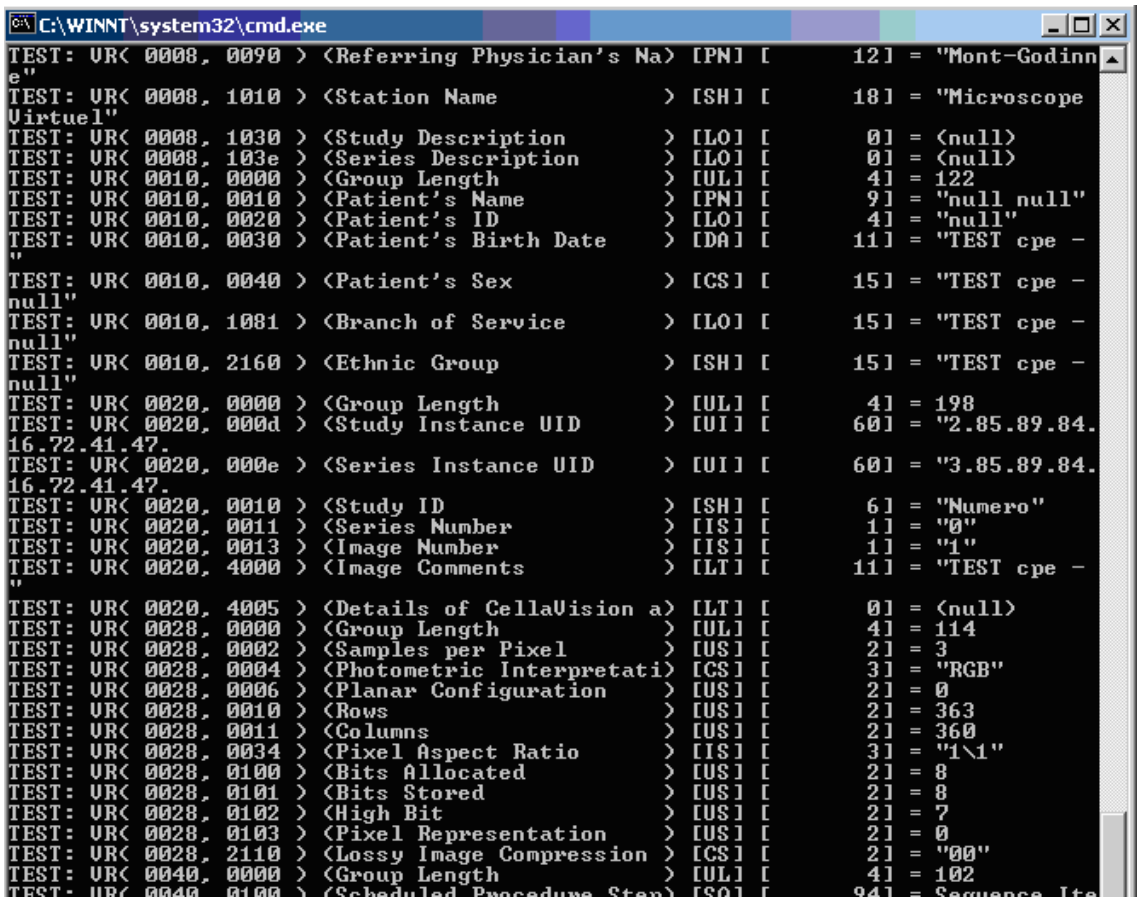

FIG. 6.13 – Console MS-DOS permettant de visualiser les logs du programme

Pour lancer l'extracteur sous forme de service (cette application n'est pas portable, elle n'a été testée que sous environnement windows) :

– Double cliquez sur " Run ExtractorService.bat "

– L'application démarre et une icône "PLAY" verte apparaît dans le systemTray de windows

FIG. 6.14 – Etat du "System Tray" Windows après le démarrage de l'extracteur

– Sélectionnez le dossier à scanner via la fenêtre de configuration disponible en cliquant sur "Configuration" dans le menu de l'icône du systemTray.

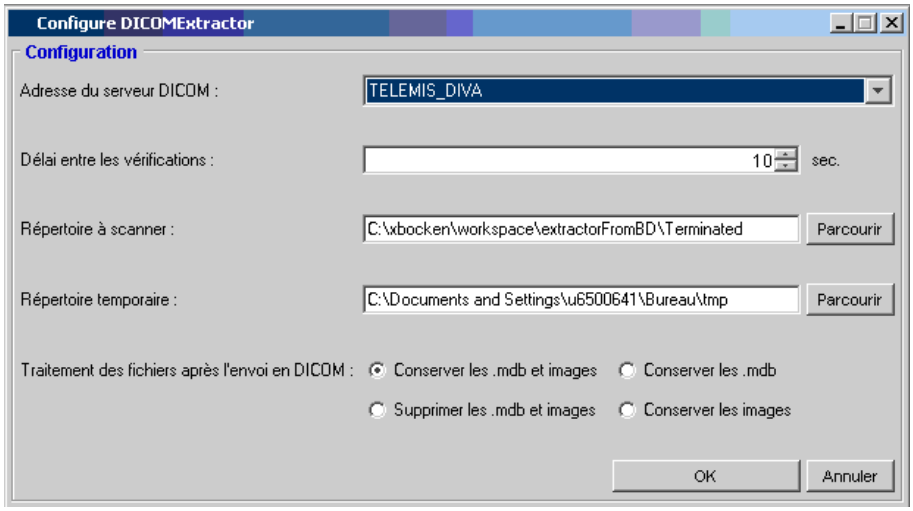

FIG. 6.15 – Fenêtre de configuration du serveur d'envoi des images

– Les configurations vous permettent de définir un ensemble de paramètres :
  - Le choix du serveur DICOM, (cfr aet.cfg) ;
  - Le temps entre les "check" ;
  - Le répertoire à scanner ;
  - Le répertoire de stockage temporaire ;
  - La politique d'après envoi.
– Même note que pour l'extracteur avec IHM, la fermeture de la fenêtre DICOM revient à fermer tout le programme. Attention à ne pas le fermer lors d'un envoi.

Références bibliographiques

Bibliographie

[Ang03] J. Angulo. Simplification morphologique d'images couleur par critères connectifs. *Ph.D. Thesis Ecole des Mines de Paris*, 2003.

[Bou] Alain Boucher. In *Traitement d'images*.

[cL99] Olympus Optical co LTD. Système d'autofocus pour microscopes d'emploi général (dispositif de mise au point motorisée utilisant la méthode U-ZD). *Mode d'emploi U-AF*, 1999.

[DBR06] B. Parrein D. Barba, P. Le Callet and V. Ricordel. Traitement numérique d'images et codage. *Cours de l'université numérique ingénierie et technologie*, 2006.

[DD02] H. Deitel and P. Deitel. *Comment programmer en JAVA*. 4ème édition, Les éditions Reynald Goulet Inc., Repentigny (Québec), 2002.

[Dec04] G. Decock. Composition automatique de frottis numériques - application aux gigaimages. *Mémoire de fin d'études en informatique, FUNDP(Facultés Universitaires Notre- Dame de la Paix) Namur Belgique*, 2004.

[Des05] Antonin Descampe. Jpeg 2000. In *Université Catholique de Louvain, Telecommunications and Remote Sensing Laboratory*, 2005.

[Geo02] B. Georges. La télémicroscopie en cytologie hématologie. *Mémoire de fin d'études en informatique, FUNDP(Facultés Universitaires Notre- Dame de la Paix) Namur Belgique*, 2002.

[GM02] V. Gouet and P. Montesinos. Normalisation des images en couleur face aux changements d'illumination. In *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, volume II, pages 415–424, Angers, France, 2002.

[Haz03] V. Hazebroucq. Rapport sur l'état des lieux, en 2003, de la télémédecine française. *Rapport établi à la demande de Madame la Ministre déléguée à la recherche et aux nouvelles technologies, Université Paris V - René Descartes*, 2003.

[Jad05] C. Jadoul. Imagerie Médicale - Etude et développement d'un système de télémicroscopie virtuel. *Mémoire de fin d'études en informatique,*

FUNDP(Facultés Universitaires Notre- Dame de la Paix) Namur Belgique, 2005.

[Man05] Laurent Manyri. *Analyse automatique d'images de populations microbiennes.* Institut National des Sciences Appliquées de Toulouse, 2005.

[MDGW07] Raphaël Marée, Marie Dumont, Pierre Geurts, and Louis Wehenkel. Random subwindows and randomized trees for image retrieval, classification, and annotation, 2007.

[MGW06] Raphaël Marée, Pierre Geurts, and Louis Wehenkel. Biological image classification with random subwindows and extra-trees, sep 2006.

[MGW07] Raphaël Marée, Pierre Geurts, and Louis Wehenkel. Content-based image retrieval by indexing random subwindows with randomized trees, Nov 2007.

[Mor04a] R. Moreda. Frottis de sang normal : frottis, coloration mgg et photographies de champs. In *Lycée Dr Lacroix, Narbonne*, 2004.

[Mor04b] R. Moreda. Les leucocytes sur frottis sanguin coloré au mgg. In *Lycée Dr Lacroix, Narbonne*, 2004.

[Oly05] Olympus. Informations complémentaires U-IFRS/U-IFGP, carte RS232C et carte GPBI. *Manuel d'utilisation du pupitre multi-commandes U-MCB*, 2005.

[Pee07] C. Peeters. Microscopie Virtuelle - Localisation, extraction et classification de globules blancs dans un frotti sanguin. *Mémoire de fin d'études en informatique, FUNDP(Facultés Universitaires Notre- Dame de la Paix) Namur Belgique*, 2007.

[Per06] B. Permentier. Etude et développement d'un système de télémicroscopie virtuelle. *Mémoire de fin d'études en informatique, IESN(Institut d'Enseignement Supérieur de Namur) Namur Belgique*, 2006.

[Zuy03] L. Zuyderhoff. Composition automatique de frottis numériques. *Mémoire de fin d'études en informatique, FUNDP(Facultés Universitaires Notre- Dame de la Paix) Namur Belgique*, 2003.

Netographie

[CCM] CCM. Ccm - "ordinateur - les ports séries et parallèles". http://www.commentcamarche.net/pc/serie.php3.

[Cel08] CellaVision. Cellavision - wbc classification and rbc overview, 2008. http://www.cellavision.com/?id=1323&cid=1582.

[EL08] Mastere IST (98) Eric Lormeau. Localisation et epaisseur des contours des detecteurs par derivation (sobel, prewitt, roberts) ou par gradient morphologique.yte, 2008. http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/localisation_et_epaisseur3/index.html.

[PIAG08] Digital Signal Processing and University Of Oslo Image Analysis Group. Xite, 2008. http://www.ifi.uio.no/research/groups/dsb/software/xite/.

[PW03] N. Petkov and M.B. Wieling. Canny edge detector (canny filter) for image processing and computer vision, 2003. http://matlabserver.cs.rug.nl/cgi-bin/matweb.exe.

[RFW03a] A. Walker R. Fisher, S. Perkins and E. Wolfart. Canny edge detector, 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm.

[RFW03b] A. Walker R. Fisher, S. Perkins and E. Wolfart. Sobel edge detector, 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm.

[Tup] Florence Tupin. La détection des contours dans les images. http://www.tsi.enst.fr/~tupin/TDI/.

[Wik] RS232 Wikipedia. Wikipedia - rs232. http://fr.wikipedia.org/wiki/RS-232.

[Wik08a] Wikipedia. Canny edge detector, 2008. http://en.wikipedia.org/wiki/Canny_edge_detector.

[Wik08b] Wikipedia. Wikipedia - coefficient de variation, 2008. http://fr.wikipedia.org/wiki/Coefficient_de_variation.

[Wik08c] Wikipedia. Wikipedia - hématie, 2008. http://fr.wikipedia.org/wiki/H%C3%A9matie.

[Wik08d] Wikipedia. Wikipedia - leucocyte, 2008. http://fr.wikipedia.org/wiki/Leucocyte.

[Wik08e] Wikipedia. Wikipedia - plasma sanguin, 2008. http://fr.wikipedia.org/wiki/Plasma_sanguin.